LYNX™

Product Family Operating Instructions

Modular LYNX System
MicroLYNX System
Software Reference

IMS™ INTELLIGENT MOTION SYSTEMS, INC.
Excellence in Motion™

370 N. MAIN ST., PO BOX 457, MARLBOROUGH, CT 06447
PH: (860) 295-6102, FAX: (860) 295-6107
Internet: www.imshome.com, E-Mail: info@imshome.com

The information in this book has been carefully checked and is believed to be accurate; however, no responsibility is assumed for inaccuracies.

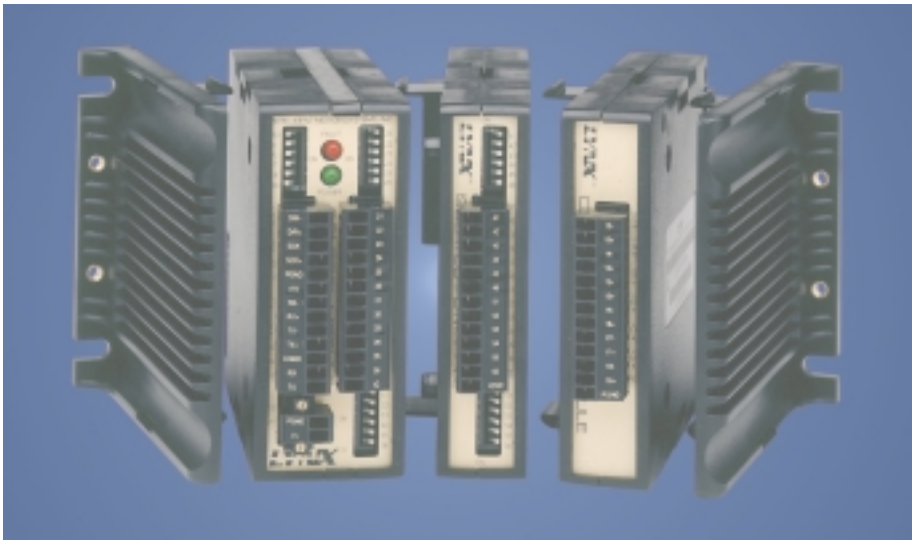
Intelligent Motion Systems, Inc., reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Intelligent Motion Systems, Inc., does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights of others. Intelligent Motion Systems and iMS™ *are trademarks of Intelligent Motion Systems, Inc.*

Intelligent Motion Systems, Inc.'s general policy does not recommend the use of its products in life support or aircraft applications wherein a failure or malfunction of the product may directly threaten life or injury. Per Intelligent Motion Systems, Inc.'s terms and conditions of sales, the user of Intelligent Motion Systems, Inc., products in life support or aircraft applications assumes all risks of such use and indemnifies Intelligent Motion Systems, Inc., against all damages.

© 2000 Intelligent Motion Systems, Inc.
All Rights Reserved

Modular LYNX

Part I
The Modular LYNX System



Getting Started
Connecting the LYNX System
Mounting the LYNX System
Powering the LYNX System
The Communications Interface
Configuring the Digital I/O
The LYNX Control Module
The LYNX Control Module (Combination)
The Isolated Digital I/O Module
The Differential Digital I/O Module
The Combination Digital I/O Module

Table of Contents

Getting Started 1-5
Section Overview 1-5
Getting Started 1-5
Included in the Package 1-5
User Provided Tools and Equipment Needed 1-6
Connecting the Power Supply 1-6
Connecting the Step Motor Driver 1-6
Motor Connections 1-6
Communications Wiring 1-6
Establishing Communications using the IMS LYNX Terminal 1-6
Testing the LYNX Setup 1-7
Connecting the LYNX System 1-9
Section Overview 1-9
Connecting the System 1-9
Mounting the LYNX System 1-10
Section Overview 1-10
Panel Mount 1-10
Din Rail Mounting Option 1-10
Included in the DIN Rail Mounting Kit 1-10
Mounting the LYNX System to a DIN Rail 1-10
Powering the LYNX System 1-12
Section Overview 1-12
Wiring and Shielding 1-12
Rules of Wiring 1-12
Rules of Shielding 1-12
LYNX Control Module with IMS Driver 1-13
Stand-alone or with Optional I/O Modules 1-14
+12 to +75VDC Supply 1-14
+5 VDC Supply 1-14
Power Requirements 1-15
The Communications Interface 1-16
Section Overview 1-16
Connecting the RS-232 Interface 1-16
Single Control Module System 1-16
Multiple Control Module System 1-17
Connecting the RS-485 Interface 1-20
Single Controller System 1-20
Multiple Controller System 1-21
LYNX Control Module Modes of Operation 1-23
Immediate Mode 1-23
Program Mode 1-23
EXEC Mode 1-23
LYNX Control Module Communication Modes 1-23
ASCII 1-23
Binary 1-24
Configuring the Digital I/O 1-25
Section Overview 1-25
System I/O Availability by Module 1-25
The Isolated Digital I/O 1-26
Uses of the Isolated Digital I/O 1-26
The IOS Variable 1-27
Configuring an Input 1-28
Configuring the Digital Filtering 1-28
Configuring an Output 1-29

The IO Variable 1-29
Read/Write an I/O Group 1-30
The Differential I/O 1-31
The Clock Interface 1-31
Clock Types Defined 1-31
Configuring The Differential I/O - The IOS Variable 1-32
Configuring an Input 1-32
Setting the Digital Input Filtering for the Differential I/O 1-33
Configuring an Output 1-33
Typical Functions of the Differential I/O 1-34
Connecting and Using an Encoder 1-34
Translating the EUNIT Variable to a Dimension of Distance 1-35
Half Axis Operation (Follower) 1-36
One and a Half Axis Operation (RATIOE) 1-37
The LYNX Control Module (LX-CM100-000) 1-39
Section Overview 1-39
Hardware Specifications 1-39
Environmental Specifications 1-39
Mechanical Specification 1-39
Connection Overview 1-40
Power Requirements 1-40
LED Indicators 1-41
Pin Assignment and Description 1-41
Switch Assignments 1-42
The LYNX Control Module (Combination) 1-43
Section Overview 1-43
Hardware Specifications 1-43
Environmental Specifications 1-43
Mechanical Specification 1-43
Connection Overview 1-44
Power Requirements 1-44
LED Indicators 1-45
Pin Assignment and Description 1-45
Switch Assignments 1-46
The Isolated Digital I/O Module 1-47
Section Overview 1-47
Hardware Specifications 1-47
Environmental Specification 1-47
Mechanical Specification 1-47
Connection Overview 1-48
Pin Assignments And Description 1-48
Switch Assignments And Description 1-49
Input Specifications 1-49
Input Filtering 1-50
Output Specifications 1-50
The Differential Digital I/O Module 1-51
Section Overview 1-51
Hardware Specifications 1-51
Environmental Specification 1-51
Mechanical Specification 1-51
Connection Overview 1-52
Power Requirements 1-52
Pin Assingments And Description 1-53
Input Specifications 1-53
Input Filtering 1-54
Output Specifications 1-55

SECTION 1

Getting Started

Section Overview

The purpose of this section is to get you up and running quickly. This section will help you do the following:

- Connect power to the LYNX Control Module.
- Connect and establish communications in single mode.
- Write a simple test program.

Getting Started

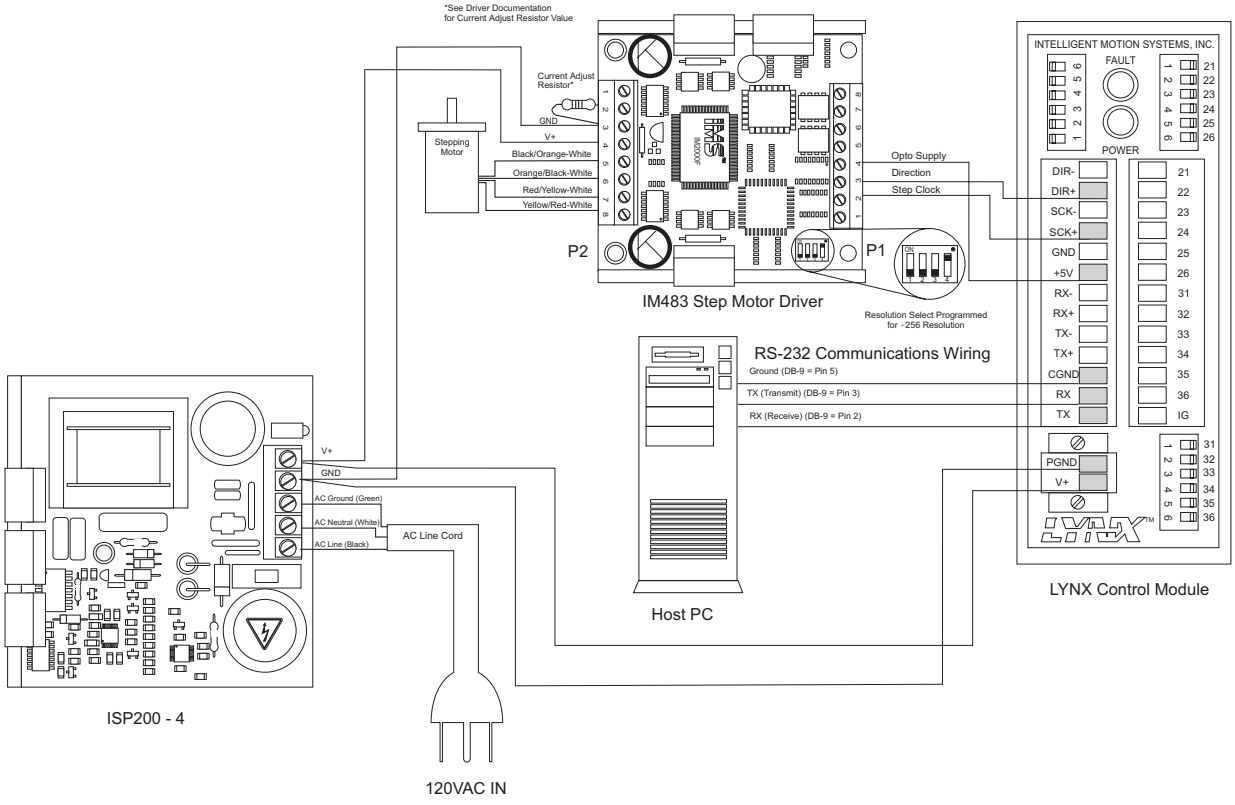


Figure 1.1: Basic Setup Configuration, RS-232 Interface

Included in the Package

(1) LYNX Control Module ..... (IMS P/N LX-CM100-000)
(2) End Mounting Brackets ..... (IMS P/N LX-EB100-000)
(1) LYNX Compact Disc ..... (IMS P/N LX-SW100-000)
(1) Screw Driver ..... (IMS P/N SD1)

User Provided Tools and Equipment Needed

- Serial Cable
- IM483 or equivalent step motor driver
- ISP200-4 or equivalent power supply
- M2-22XX or equivalent stepping motor
- Wire Cutters/Strippers
- 22 gauge wire for logic level signals
- 18 gauge wire for power supply and motor wiring
- PC with a free serial port (COM 1 or 2)

Connecting the Power Supply

1. Using the 18 gauge wire, connect the DC output of your power supply to V+ on your LYNX Control Module, and to P2, pin 4 on the IM483 Step Motor Driver. (Or V+ pin on equivalent driver.) *Figure 1.1.*
2. Connect the Power Supply Return (GND) to PGND on the LYNX Control Module, and to P2, pin 3 on the IM483 Step Motor Driver. (Or GND on equivalent driver.) *Figure 1.1.*
3. Connect the AC Line cord to your power supply in accordance with any user documentation accompanying the supply. **DO NOT PLUG IN AT THIS TIME!**

Connecting the Step Motor Driver

1. Using 22 gauge wire, connect direction DIR+ on the LYNX Control Module to P1, pin 3 on the IM483 Driver. (Or direction pin of equivalent drive used.) *Figure 1.1.*
2. Connect Step Clock SCK+ of the LYNX Control Module to P1, pin 2 of the IM483 Driver. (Or Step Clock input of equivalent drive used.) *Figure 1.1.*
3. Connect the +5V output off the LYNX Control Module to the Opto Supply P1, pin 4 of the IM483 Driver. (Or Opto Supply of drive used if required.) *Figure 1.1.*
4. Set the Resolution Select DIP switch on the IM483 Driver to ÷256 resolution. *Figure 1.1.*

Motor Connections

Connect the motor to the IM483 Step Motor Driver in accordance with *Figure 1.1.*

Communications Wiring

Connect the Host PC to the LYNX Control Module (RS-232 Communications) in accordance with *Figure 1.1.* This is needed to program the LYNX Control Module.

Establishing Communications using the IMS LYNX Terminal

Included in the LYNX shipping package is the IMS LYNX Terminal software. This is a programming/ communications interface created by IMS to simplify the use of the MicroLYNX. There is a 32 bit version for Windows 9x/NT4/2000 located on the CD. The IMS LYNX Terminal is also necessary to upgrade the software in your LYNX Control Module. These updates will be posted to the IMS website at http:// www.imshome.com/ as they are made available.

To install the IMS LYNX Terminal to your hard drive, insert the CD into your CD-ROM Drive. The 3.5" CD, while smaller than typical compact disks, will work in any tray-type CD drive. To install click "Start > Run" and type "x:\terminal\32bit\setup.exe" in the "Open" box.

Follow the on-screen instructions to complete the installation.

1) Open the LYNX Terminal by selecting Start>Programs>LynxTerm>LYNXTERM (Windows 9x/ NT/2000).

2) Click the File Menu Item “Setup”.
3) Select the “Terminal>Setup” option.
4) Select the Communications Port that you will be using with your MicroLYNX.
5) The BAUD rate is already set to the MicroLYNX default. Do not change this setting until you have established communications with the MicroLYNX Controller.
6) The “Window Size” settings are strictly optional. You may set these to whatever size is comfortable to you.
7) Click “OK”. The settings will be automatically saved upon a normal shutdown.
8) Apply power to the MicroLYNX Controller. The following sign-on message should appear in the Terminal window:

```
Program Copyright © 1996-2000 by:
Intelligent Motion Systems, Inc.
Marlborough, CT 06447
VER =       SER =
```

Detailed instructions for the IMS LYNX Terminal software can be located in Part III: Using the LYNX Terminal Software, of this manual.

Testing the LYNX Setup

Two basic instructions for communicating with a control module are SET and PRINT. The SET instruction is assumed and can be left off when communicating in ASCII mode. (You are in ASCII mode whenever you are using a text based terminal). It is used to set variables and flags that define control module operation. The LYNX Software automatically recognizes the SET instruction whenever the name of the variable or flag is typed into the terminal. Here we will set the motor units variable (MUNIT) to 51200 by typing the following at the prompt (>):

```
MUNIT = 51200
```

The PRINT instruction is used to report the values of variables and flags. Now, double-check the value of MUNIT by typing the following at the prompt (>):

```
PRINT MUNIT
```

The return from your terminal should be 51200. Note that the case is not important for instructions, variables, and flags. They may be typed in upper or lower case.

Use the SLEW instruction to move the motor at a constant velocity. Be sure that the velocity provided is a reasonable value for your motor and drive and try to move the motor. For instance, at the prompt type:

```
SLEW 10
```

This will move the motor at a speed of 10 munits per second. If the motor does not move, verify that the wiring is in accordance with *Figure 1.1* and that the resolution select settings agree with part 5, step 4 on the other side of this page. If a non IMS driver is being used, you may need to consult the user manual for that device.

Once you have been able to move the motor, the next step is to write a simple program to illustrate one of the dynamic features of the LYNX: the ability to convert motor steps to a dimension of linear or rotary distance. Let’s begin by discussing the relationship between the MUNIT variable and user units. Typically when we perform a move we want to know the distance of that move in a familiar unit of measurement. That means translating motor steps to the desired unit of measurement. The LYNX Control Module has the capability of doing this for you. You have already set the motor units variable (MUNIT) to a value 51,200. With the driver set to a resolution of 256 micro-steps per step and a 1.8° step motor that will be equal to 1 revolution of the motor, or one USER UNIT. A user unit can be any unit of measure. At this point, by entering the instruction MOVR 1, the motor will turn one complete revolution relative to it’s current position. Therefore, 1

User Unit = 1 Motor Revolution. For the exercise below we will use degrees for our user unit. As the LYNX Product Manual indicates, the calculation required to select degrees as our user unit in this case is:

51200 Micro-steps per rev ÷ 360 degrees = 142.222 Micro-steps per degree

By setting the MUNIT variable to 51200/360 the LYNX Control Module will perform the calculation to convert the user unit to degrees. Now, when issued, a relative motion instruction "MOVR 90" the motor will turn 90 degrees.

Now, enter a sample program that will convert motor steps to degrees, execute a 90° move, and report that move every 100 milliseconds while the motor is moving. Type the following bold commands:

```
'Enter Program Mode, start program at Location 2000.
PGM 2000
'Label the program TSTPGM.
LBL TSTPGM
' Set the user units to degrees.
MUNIT = 51200/360
' Set the max. velocity to 25 degrees per second.
VM = 25
' Execute a relative move of 90 degrees.
MOVR 90
' Report the position every 100 ms while moving.
LBL PRINTPOS
DELAY 100
PRINT "Axis position is", POS, "Degrees."
BR PRINTPOS, MVG
'End the program.
END
PGM
```

Now Type **TSTPGM** to run program.
This sample program will be stored starting at location 2000. It sets the conversion factor for the user units, sets the maximum velocity and then starts a motion. While the motion is occurring, the position is reported every 100 milliseconds.

At this point you may desire to restore the settings to their factory default as you may not wish to use degrees as your user unit. To do this, you will use the CP, DVF, and IP instructions.

CP - Clear Program.
To clear the program, type CP 1, 1. This will completely clear program memory space. Should you desire to only remove one program, the instruction "CP [Program Label]" i.e., "CP TSTPGM" would clear only that program. In this exercise only one program was entered, "CP TSTPGM" will clear it.

DVF - Delete User Defined Variables and Flags.

By entering DVF, all of the user defined variables will be removed. Although no Flags were set in this exercise, this command would clear them were they used.

IP - Initialize Parameters

This instruction will restore all of the parameters to their factory default state.

After entering these instructions a SAVE instruction should be entered.

SECTION 2

Connecting the LYNX System

Section Overview

Each module of the LYNX System is a closed unit with a header of pins and locking tabs to connect it to another module in the system. Optional I/O modules are connected on the RIGHT side of the Control Module. This section covers:

- Removing the End Plates.
- Connecting/Disconnecting System Modules.

Connecting the System

1. Remove the end plate(s) [A] from the Control Module. Depressing the locking clips [C] with a small screwdriver through the slot [B] on the top and bottom of the module and pulling them apart does this. See figure 2.1
2. Align the locking clips of the module being connected with the slots on the module being connected to.
3. Press modules firmly together, there will be an audible "snap" when the locking clips are fully engaged.
4. Reinstall the end plates at the ends of the LYNX System. They are designed to fit either end.
5. You are now ready to mount your LYNX System to a panel or DIN Rail using the optional hardware kit.

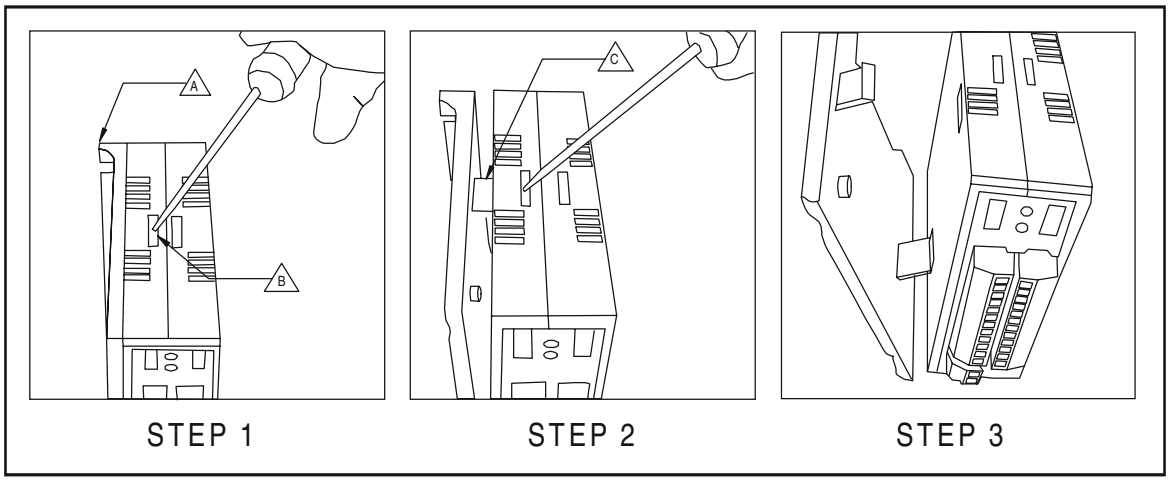


Figure 2.1: Removing the End Plates

WARNING! Exercise caution when removing end plates or separating LYNX System modules! Internal component damage may occurif the screwdriver is inserted too far into the slots!

SECTION 3

Mounting the LYNX System

Section Overview

This section covers the two basic methods of mounting the LYNX System.

- Panel Mount.
- DIN Rail Mounting Option.

Panel Mount

Using the panel mount option, the LYNX is designed to use #10 hardware (not included). Details such as screw length and threads are dependent on your overall system design.

Din Rail Mounting Option

A DIN Rail mounting kit (IMS P/N LX-DB100-000) may be purchased as an option to your LYNX System. It includes all the hardware necessary to mount the system to either of the following recommended DIN rails:

TS35 X 7.5 or TS35 X 15

Included in the DIN Rail Mounting Kit

Included in the DIN Rail Mounting Kit is the following hardware:

- 2 - IMS0065 DIN Rail Brackets
- 4 - #6 Split Lock Washer
- 4 - #6-32X7/16 L Pan Hd Machine Screws
- 4 - #6 Flat Washer .040 Thick
- 2 - #6 X .250 L Set Screw
- 1 - Instruction Sheet

Mounting the LYNX System to a DIN Rail

In order to install your LYNX System on a DIN rail complete the following:

1. Insert the two DIN rail brackets into the slots located in the back of the system between the end plates and LYNX modules. The pull-tab on the DIN rail bracket must be on the bottom.

2. Using the #6 hardware provided, secure the bracket to the end plates. *Figure 3.1.* Tighten to 5 - 7 lb/in.

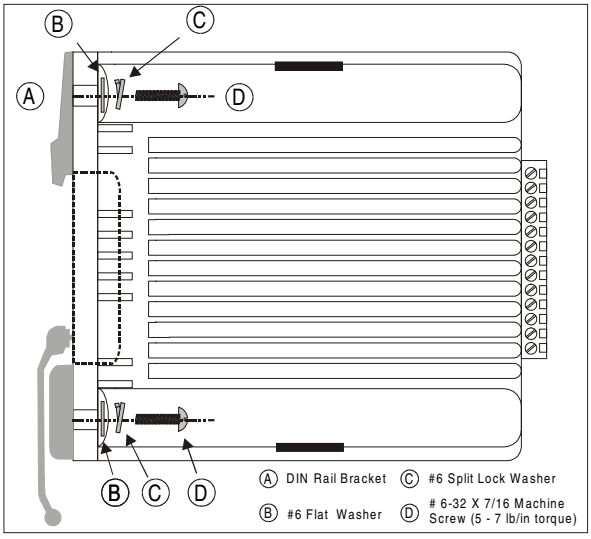


Figure 3.1: Installing the DIN Rail Bracket

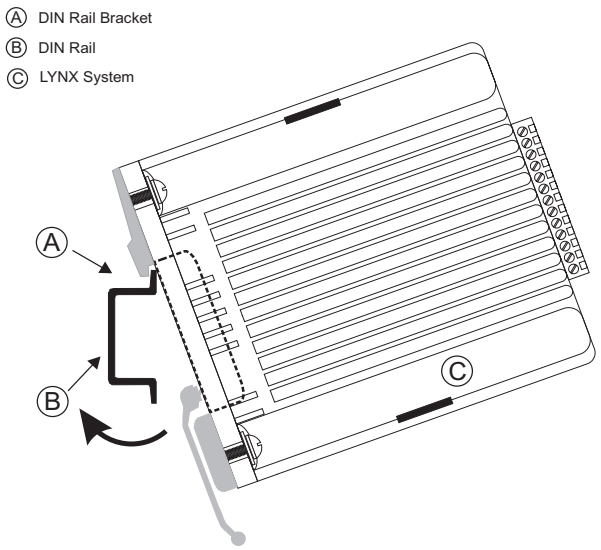


Figure 3.2: Installing the LYNX System on a DIN Rail

3. Holding the LYNX System at an angle away from you, lower the upper slot of the DIN rail attachment onto the top edge of the DIN rail. Snap LYNX system into place. *Figure 3.2.*

4. Insert #6 X .250 L set screw (provided) into the TOP threaded insert located between the #6 screws on each end plate. *Figure 3.3.* Tighten until 12-14 in/oz. This will keep the system from sliding on the DIN rail.

To Remove the LYNX System from the DIN Rail:

1. Loosen the set screws located in the TOP threaded insert between the #6 screws on each end plate.
2. Grasp the pull-tabs located on the bottom of the DIN Rail brackets to release the LYNX system from the DIN Rail *(Figure 3.3 - C&D)* while gently lifting the front of the LYNX system.
3. Lift the LYNX System Away from the DIN Rail.

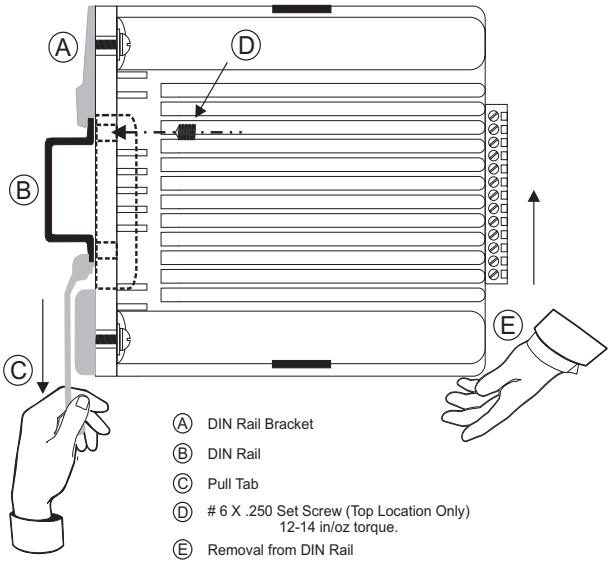


Figure 3.3: Installing the Set Screw, Removing the LYNX System from the DIN Rail

NOTE: The DIN Rail Mounting option should only be used on STATIONARY Systems. It is not designed for transport!

SECTION 4

Powering the LYNX System

Section Overview

This section covers the two basic power configurations for your LYNX System.

- Basic rules of wiring and shielding.
- LYNX Control Module with IMS Drivers.
- LYNX Control Module as Stand-alone or with Optional I/O Module.

Wiring and Shielding

Noise is always present in a system that involves high power and small signal circuitry. Regardless of the power configuration that you use in your system, there are some wiring and shielding rules that you should follow to keep your noise-to-signal ratio as small as possible.

Rules of Wiring

- Power Supply and Motor wiring should be shielded twisted pairs run separately from signal carrying wires.
- A minimum of 1 twist per inch is recommended.
- Motors wiring should be shielded twisted pairs using 20-gauge wire, or 18 gauge or better for distance greater than 5 feet.
- Power ground return should be as short as possible to established ground.
- Power Supply wiring should be shielded twisted pairs. Use 18 Gauge wire if load is less than 4 amps, or 16 gauge for more than 4 amps.
- Do not “Daisy-Chain” power wiring to system components.

Rules of Shielding

- The shield must be tied to zero-signal reference potential. In order for shielding to be effective it is necessary for the signal to be earthed or grounded.
- Do not assume that earth ground is true earth ground. Depending on the distance to the main power cabinet it may be necessary to sink a ground rod at a critical location.
- The shield must be connected so that shield currents drain to signal-earth connections.
- The number of separate shields required in a system is equal to the number of independent signals being processed plus one for each power entrance.
- The shield should be tied to a single point to prevent ground loops.
- A second shield can be used over the primary shield, however the second shield is tied to ground at both ends.

WARNING! When using an unregulated supply, ensure that the output voltage does not exceed the maximum driver input voltage due to variations in line voltage! It is recommended that an input line filter be used on power supply to limit voltage spikes to the system!

LYNX Control Module with IMS Driver

In this case, power is connected to the LYNX Control Module via connector P1. All optional plug on modules are then powered from the LYNX Control Module. In this configuration pins 5 and 6 on connector P2 of the Control Module becomes a +5VDC (150mA, internally limited) regulated output. If an encoder is to be used in the system, it may be powered via these pins. Below is a table of recommended power supply specifications for each IMS drive.

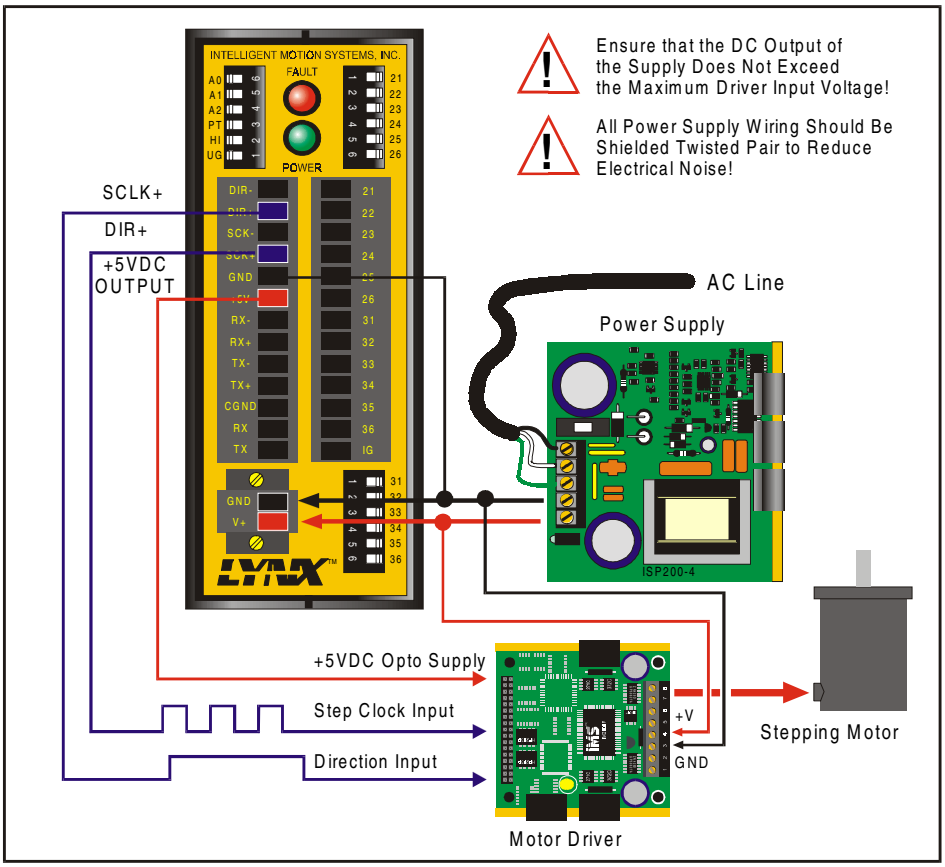


Figure 4.1: Power Configuration. LYNX Control Module and external IMS Driver

Power Supply Recommendations	
Recomended Type	Unregulated DC
Ripple Voltage	±10%
When Used With IM483/IM483H	
Output Voltage	+12 to +45VDC
*Output Current	2A (Typ.) 4A (Peak)
When Used With IM804/IM805/IM805H	
Output Voltage	+24 to +75VDC
*Output Current	4A (Typ.) 6A (Peak)

*The output current needed is dependant on the supply voltage, motor selection and load.

+12 to +75VDC Supply

A +12 to +75VDC unregulated supply connected to P1 provides power to the LYNX Control Module and any optional I/O modules. As in the LYNX Controller with Driver (s) Configuration, pins 5 (Ground) and 6 (+5VDC) on connector P2 of the Control Module becomes a +5VDC (150mA, internally limited) regulated output.

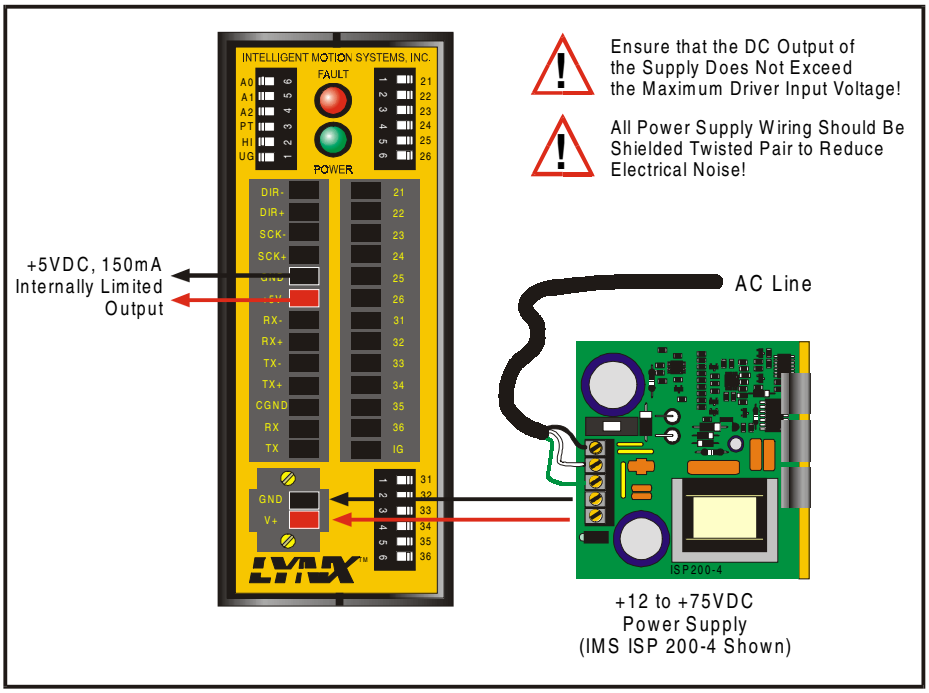


Figure 4.2: Stand-alone Power Configuration: 12 - 75 VDC Supply

+5 VDC Supply

A +5VDC ±5% regulated supply connected to pins 5 (Ground) and 6 (+5VDC) on connector P2 provides power to the LYNX Control Module and any optional I/O modules. *Figure 4.3.* It is assumed that external drives are being used and power is supplied to these drives separately. The LYNX Controller internally limits the current to 800mA. While the LYNX Controller and I/O Modules will only require 368mA, a fully configured LYNX System utilizing the outputs may require up to 800mA.

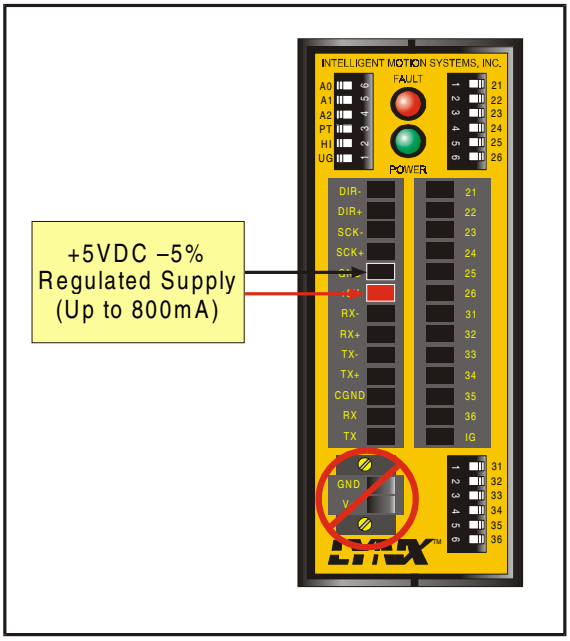


Figure 4.3: Stand-alone Power Configuration: 5 VDC

Power Requirements

Power Requirements and Specifications	
Input Voltage	+12 to +75 VDC Unregulated or +5VDC ±5%
Input Current	250mA (5VDC input) 165mA (+12VDC Input)* 95.0mA (+48 VDC Input)* 84.5mA (+75VDC Input)* *I/O and +5VDC output unloaded (Control Module Only)
Output Voltage	+5VDC ±5%
Output Current	150mA (Internally Limited

Input Current Requirements per Module	
LYNX Control Module	250 mA (+5VDC Input)
Isolated Digital I/O Module	68mA (5VDC Input)
Differential I/O Module	50mA (+5VDC Input)
Output Current	150mA (Internally Limited

Table 4.1: Power Requirements

WARNING! When using an unregulated supply, ensure that the output voltage does not exceed the maximum driver input voltage due to variations in line voltage! It is recommended that an input line filter be used on power supply to limit voltage spikes to the system!

WARNING! When specifying the input voltage of the LYNX System ensure that the power supply output voltage corresponds with the input voltage of the driver used!

WARNING! When specifying an external power supply ensure that all modules are included in the power calculation!

WARNING! Only one of these methods of Powering the LYNX System can be used!

SECTION 5

The Communications Interface

Section Overview

The LYNX Control Module features two communication interfaces: RS-232 and RS-485. For both channels, the BAUD rate is software configurated, using the BAUD variable, to 4800, 9600, 19200 or 38400 bits/sec. The factory default is set to 19200 bits/sec. Default data settings are 8 data bits, 1 stop bit and no parity.

A host computer can be connected to either interface to provide commands to the control module or to multiple control modules in a system. Since most personal computers are equipped with an RS-232 serial port, it is most common to use the RS-232 interface for communications from the host computer to the control module. You will typically want to use this interface option if your Host PC will be within 50 feet of your system. Should your system design place the LYNX Control Module at a distance greater than 50 feet, it will be necessary for you to use the RS-485 interface option. You can accomplish this by using either an RS-232 to RS-485 converter, such as the converter sold by IMS (Part # CV-3222), or installing an RS-485 board in an open slot in your host PC.

Covered in detail in this section are:

- RS-232 Interface, Single Control Module System.
- RS-232 Interface, Multiple Controller System.
- RS-485 Interface, Single Control Module Interface.
- RS-485 interface, Multiple Controller System.
- Communicating with the LYNX System using Windows95/98 HyperTerminal.
- Communicating with The LYNX System using the LYNX Terminal software.
- LYNX Control Module Modes of Operation.
- LYNX Control Module Communication Modes.

Connecting the RS-232 Interface

Single Control Module System

In systems with a single control module, also referred to as Single Mode, the LYNX Control Module is connected directly to a free serial port of the Host PC. Wiring and connection should be performed in accordance with the following table and diagram. In this mode the PARTY switch will be in the OFF position, and the PARTY Flag will be set to 0 in software. This is the factory default setting. Please be aware that you cannot communicate with the LYNX Control Module in single mode unless those conditions exist.

WARNING! Failure to connect communications ground as shown may result in damage to the Control Module and/or Host!

NOTE: If using the RS-232 Interface Option, the Host PC MUST be less than 50 feet from the Control Module. If your system will be greater than 50 feet from the Host PC you must use the RS-485/RS485 Interface..

RS-232 Interface: Wiring And Connections					
LYNX Control Module		25 Pin Serial Port on PC		9 Pin Serial Port on PC	
Recieve Data (RX)	Pin 12	Transmit Data (TX)	Pin 2	Transmit Data (TX)	Pin 3
Transmit Data (TX)	Pin 13	Recieve Data (RX)	Pin 3	Recieve Data (RX)	Pin 2
Communications Ground	Pin 11	Communications Ground	Pin 7	Communications Ground	Pin 5

Table 5.1: Wiring Connections: RS-232 Interface Single Control Module System

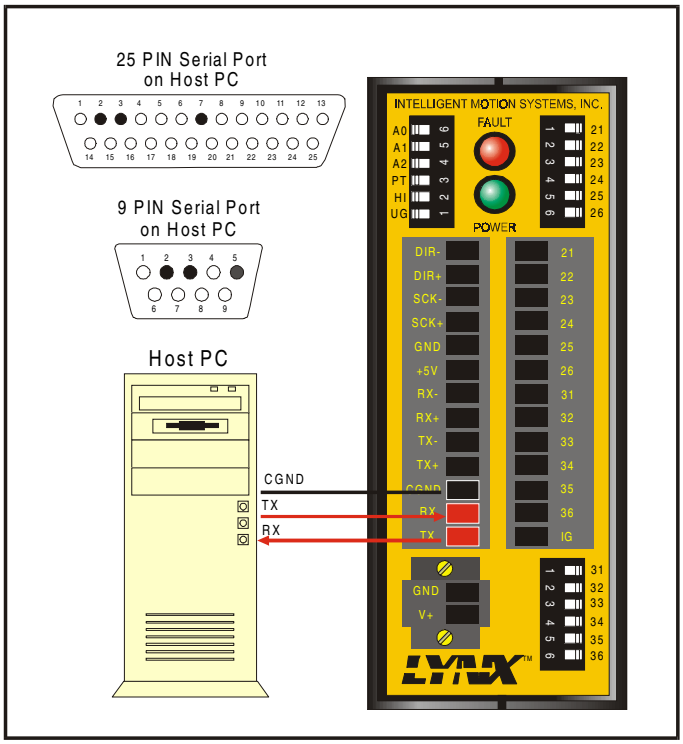


Figure 5.1: Connecting the RS-232 Interface, Single Control Module System

Multiple Control Module System

When connecting multiple control modules in a system using the RS-232 interface it is necessary to establish one control module as the HOST. This control module will be connected to the Host PC exactly as the system using a single control module. The system HOST is established by one of two methods, by manually selecting the Host switch (configuration switch #2, labeled HI) to the ON position, or, by setting the HOST Flag to True (1) in software. The remaining control modules in the system must then be connected to the HOST control module using the RS-485 interface and will have their Host switch set to OFF (HOST Flag = 0).

In this interface configuration Host PC communications will be received by the Host Control Module via RS-232 and forwarded to all of the other control modules in the system via the RS-485 channel. Responses from the individual control modules in the system will be routed back to the Host Control Module via the RS-485 channel, then internally converted to RS-232 before being forwarded back to the Host PC.

In systems with multiple controllers it is necessary to communicate with the control modules using PARTY Mode of operation. The LYNX Control Modules in the system are configured for this mode of operation by setting the Party Switch (configuration switch #3, labeled PT) to the ON position, or setting the PARTY Flag to True (1), in software. It is necessary for all of the controllers in a system to have this configuration selected. When operating in PARTY Mode each control module in the system will need a unique address, or

name, to identify it in the system. This can be done using configuration switches A0-A2, or by using the software command SET DN. For example, to set the name of a controller to "A" you would use the following command: SET DN = "A". The factory default name is "!". To set the address of the controller using the configuration switches use the following table:

Party Mode Address Configuration Switches			
Address	**A2**	**A1**	**A0**
None	OFF	OFF	OFF
A	OFF	OFF	ON
B	OFF	ON	OFF
C	OFF	ON	ON
D	ON	OFF	OFF
E	ON	OFF	ON
F	ON	ON	OFF
G	ON	ON	ON

Table 5.2: Party Mode Address Configuration Switch Settings

In setting up your system for PARTY operation the most practical approach would be to observe the following steps:

1. Connect the Host Control Module to the Host PC configured for single mode operation.
2. Establish communications with the HOST Control Module. (For help in doing this see Software Reference: Using the LYNX Terminal.) Using the Command: SET DN or the configuration switches, give the controller a unique name. If using the software command this can be any upper or lower case ASCII character or number 0-9. Save the name using the command SAVE.
3. Set the appropriate HOST and PARTY configuration in accordance with the table and diagram below. Remove power.
4. Connect the next control module in the system in accordance with the following table and diagram, setting the PARTY switch in the ON position. If you desire you can set the PARTY Flag to “1” in software later and turn the switch off.
5. Establish communications with this module using the factory default name “!”. This name cannot be reused. Rename and save the new name. Remove power.
6. Repeat the last two steps for each additional control module in the system.

WARNING! Failure to connect communications ground as shown may result in damage to the Control Module and/or Host!

NOTE: If using the RS-232 Interface Option, the Host PC MUST be less than 50 feet from the Control Module. If your system will be greater than 50 feet from the Host PC you must use the RS-485/RS485 Interface..

RS-232 Interface: Wiring And Connectionsfor Multiple LYNX Nodes					
Host Control Module		Control Module #1		Control Module #*n*	
Recieve Data (RX-)	Pin 7	Transmit Data (TX-)	Pin 9	Transmit Data (TX-)	Pin 9
Recieve Data (RX+)	Pin 8	Transmit Data (TX+)	Pin 10	Transmit Data (TX+)	Pin 10
Transmit Data (TX-)	Pin 9	Recieve Data (RX-)	Pin 7	Recieve Data (RX-)	Pin 7
Transmit Data (TX+)	Pin 10	Recieve Data (RX+)	Pin 8	Recieve Data (RX+)	Pin 8
Communications Ground	Pin 11	Communications Ground	Pin 11	Communications Ground	Pin 11
HOST Switch = ON or HOST Flag = TRUE (1)		HOST Switch = OFF or HOST Flag = FALSE (0)		HOST Switch = OFF or HOST Flag = FALSE (0)	
PARTY Switch = ON or PARTY Flag = TRUE (1)		PARTY Switch = ON or PARTY Flag = TRUE (1)		PARTY Switch = ON or PARTY Flag = TRUE (1)	

Table 5.3: Connections and Settings Multiple Control Module System, RS-232 Interface

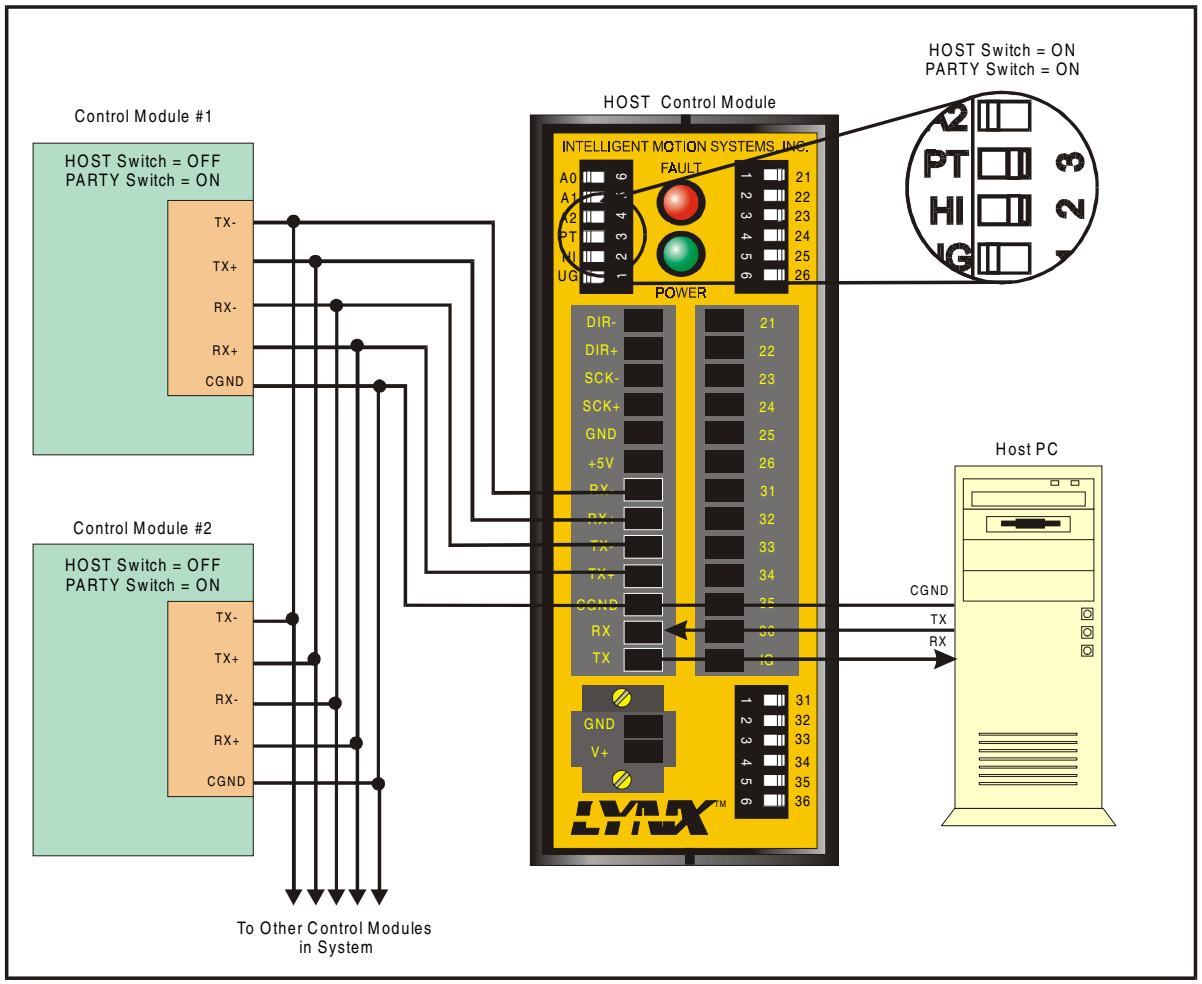


Figure 5.2: RS-232 Interface, Multiple Control Module System

Connecting the RS-485 Interface

Single Controller System

In a Single Controller System, the RS-485 interface option would be used if the Control Module is located at a distance greater than 50 feet from the Host PC. Since most PC's do not come with an RS-485 board pre-installed, you will have to install an RS-485 board in an open slot in your PC, or purchase an RS-232 to RS-485 converter, such as the CV-3222 sold by IMS, to use this connection interface. For wiring and connection information please use the following table and diagram:

RS-485 Interface: Wiring And Connections		
RS-485 Board or RS232 to RS-485 Converter	LYNX Control Module	
Recieve Data (RX-)	Transmit Data (TX-)	Pin 9
Recieve Data (RX+)	Transmit Data (TX+)	Pin 10
Transmit Data (TX-)	Recieve Data (RX-)	Pin 7
Transmit Data (TX+)	Recieve Data (RX+)	Pin 8
Communications Ground	Communications Ground	Pin 11

Table 5.4: RS-485 Interface Connections

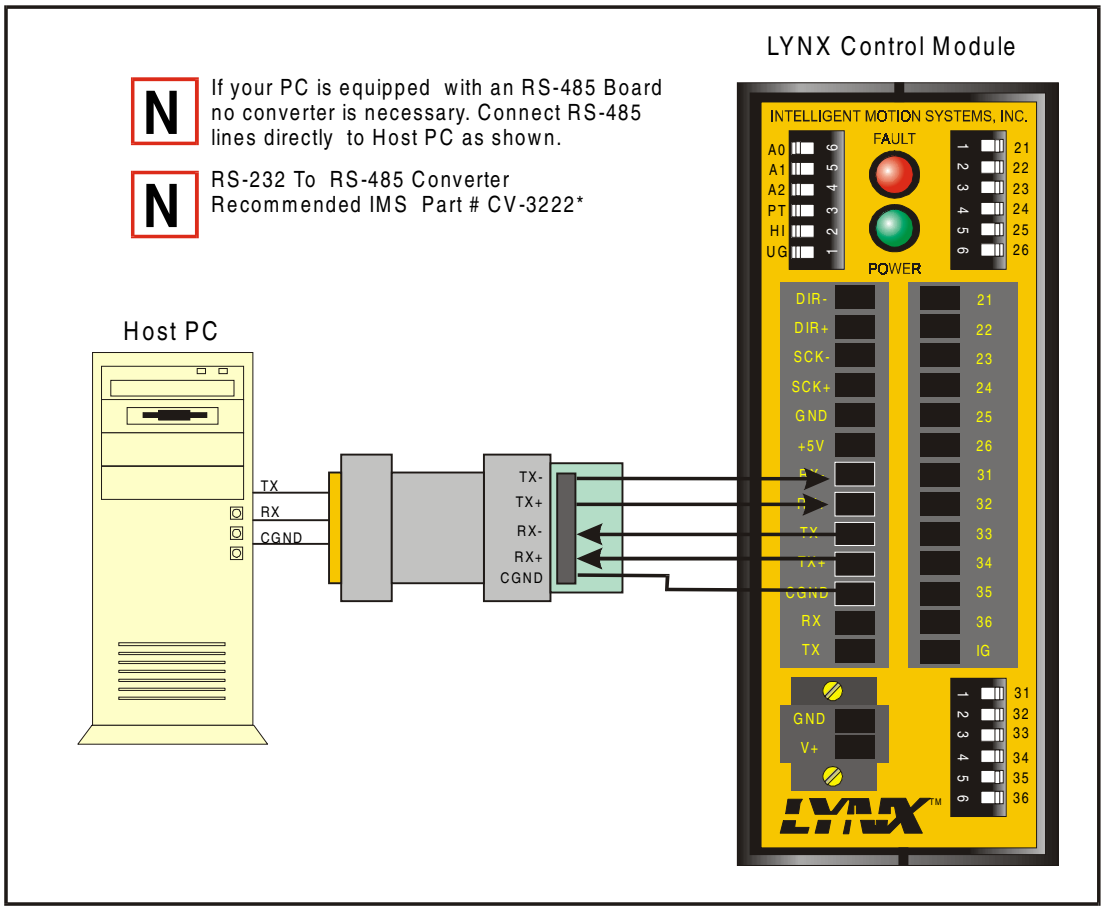


Figure 5.3: RS-485 Interface, Single Controller System

NOTE: The HOST switch MUST be off to communicate with the Control Module in a Single Controller System using the RS-485 Interface.

Multiple Controller System

When using the RS-485 interface in a Multiple Controller System, the Host PC as well as all of the control modules communicate on the RS-485 interface. In this case, there is no Host Interface Control Module, so all control modules in the system should have their Host switch OFF or HOST flag set to False (0). The Host PC will be equipped with an RS-485 board or RS-232 to 485 converter.
In systems with multiple controllers it is necessary to communicate with the control modules using PARTY Mode of operation. The LYNX Control modules in the system are configured for this mode of operation by setting the Party Switch (configuration switch #3, labeled PT) to the ON position or setting the PARTY Flag to True (1), in software. It is necessary for all of the controllers in a system to have this configuration selected. When operating in PARTY Mode each control module in the system will need a unique address, or name, to identify it in the system. This can be done using configuration switches A0-A2, or by using the software command SET DN. For example, to set the name of a controller to "A" you would use the following command: SET DN = "A". The factory default name is "!". To set the address of the controller using the configuration switches use the above table.

Party Mode Address Configuration Switches			
Address	**A2**	**A1**	**A0**
None	OFF	OFF	OFF
A	OFF	OFF	ON
B	OFF	ON	OFF
C	OFF	ON	ON
D	ON	OFF	OFF
E	ON	OFF	ON
F	ON	ON	OFF
G	ON	ON	ON

Table 5.5: Party Mode Address Configuration Switch Settings

In setting up your system for PARTY operation the most practical approach would be to observe the following steps:

1. Connect the Host Control Module to the Host PC configured for Single Mode Operation.
2. Establish communications with the HOST Control Module. (For help in doing this see: Using the LYNX Terminal in the next section.) Using the Command: SET DN or the configuration switches, give the controller a unique name. If using the software command this can be any upper or lower case ASCII character or number 0-9. Save the name using the command SAVE.
3. Set the appropriate HOST and PARTY configuration in accordance with the following table and diagram. Remove power.
4. Connect the next control module in the system in accordance with the following table and diagram, setting the PARTY switch in the ON position. If you desire you can set the PARTY Flag to "1" in software later and turn the switch off.
5. Establish communications with this module using the factory default name "!". This name cannot be reused. Rename and save the new name. Remove power.
6. Repeat the last two steps for each additional control module in the system.

RS-485 Interface: Wiring And Connectionsfor Multiple LYNX Nodes				
RS-232 to RS-485 Converter	Control Module #1		Control Module #*n*	
Recieve Data (RX-)	Transmit Data (TX-)	Pin 9	Transmit Data (TX-)	Pin 9
Recieve Data (RX+)	Transmit Data (TX+)	Pin 10	Transmit Data (TX+)	Pin 10
Transmit Data (TX-)	Recieve Data (RX-)	Pin 7	Recieve Data (RX-)	Pin 7
Transmit Data (TX+)	Recieve Data (RX+)	Pin 8	Recieve Data (RX+)	Pin 8
Communications Ground	Communications Ground	Pin 11	Communications Ground	Pin 11
	HOST Switch = OFF or HOST Flag = FALSE (0)		HOST Switch = OFF or HOST Flag = FALSE (0)	
	PARTY Switch = ON or PARTY Flag = TRUE (1)		PARTY Switch = ON or PARTY Flag = TRUE (1)	

Table 5.6: RS-485 Interface Connections and Settings, Multiple Control Module System

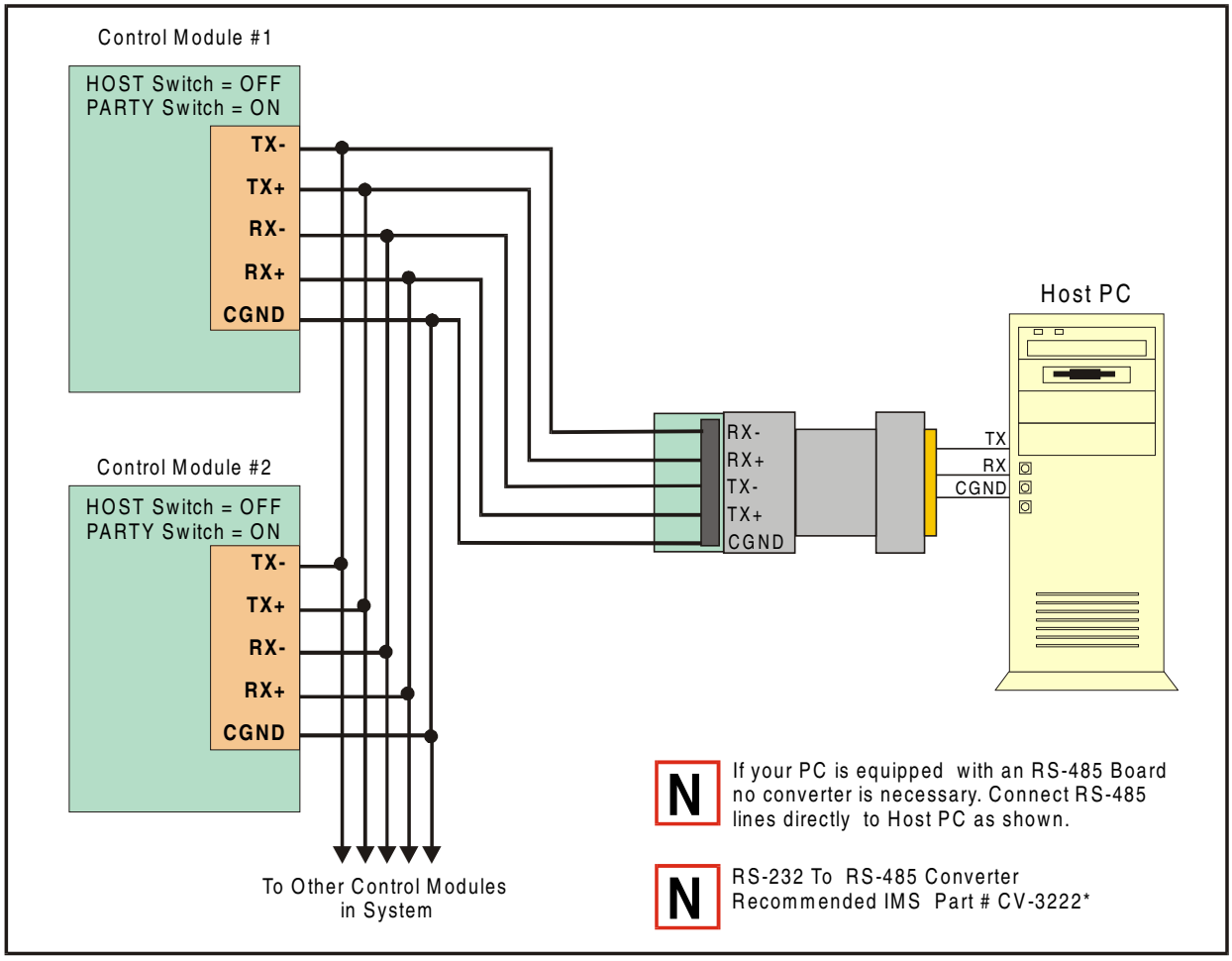


Figure 5.4: RS-485 Interface, Multiple Control Module System

It is also possible to communicate with a controller in the system in single mode by sending it a command (with address) to clear the party flag and then communicate with it as in single mode (no line feed terminator) then reset the PARTY Flag when done.

LYNX Control Module Modes of Operation

There are three modes of operation for the LYNX control module. These are Immediate Mode, Program Mode, and EXEC Mode.

Immediate Mode

In this mode, the control module responds to instructions from the user that may be a result of the user typing instructions directly into a host terminal, or of a user program running on the host which communicates with the control module.

Program Mode

The second mode of operation of the control module is Program Mode. All user programs are written in this mode. Unlike the other modes of operation, no commands or instructions can be issued to the control module in Immediate Mode. This mode is exclusively for writing programs for the controller. The command to enter Program Mode is PGM <address>. When starting Program Mode, you must specify at what address to enter the program instructions in the program space. Simply type PGM again when you have finished entering your program commands to go back to Immediate Mode.

EXEC Mode

In EXEC Mode a program is executed either in response to the EXEC instruction from the user in Immediate Mode, or in response to a specified input. While the control module is running a program, the user may still communicate with it in Immediate Mode. As part of a user program, the control module may start a second task using the RUN instruction. Thus, there can be two tasks running on the control module at the same time, a foreground task (started by the EXEC instruction in Immediate Mode) and a background task (started by the RUN instruction in Program Mode).

LYNX Control Module Communication Modes

When the control module is operating in Immediate Mode, there are two methods of communicating. The first is ASCII where the instructions are communicated to the control module in the form of ASCII mnemonics and data is also given in ASCII format. The second is binary where the instruction is in the form of an OpCode and numeric data is given in IEEE floating point hex format. In binary mode, there is also the option of including a checksum to ensure that information is received properly at the control module. The BIO flag controls the method of communication. When it is True (1), the binary method should be used and when it is False (0), the ASCII method should be used.

ASCII

ASCII is the most common mode of communicating with the LYNX System. It allows the use of readily available terminal programs such as HyperTerminal, ProComm, and the new LYNX Terminal.

When using the ASCII method of communications, the control module tests for four special characters each time a character is received. These characters are given in the table below along with an explanation of what occurs when the character is received.
The command format in ASCII mode when the control module is in Single Mode (PARTY = FALSE) is:

<Mnemonic><white space><ASCII data for 1st parameter>, <ASCII data for 2nd parameter>, … , <ASCII data for nth parameter><CR/LF>

The mnemonics for Control Module instructions, variables, flags and keywords are given in Section 6 of this document. White space is at least one space or tab character. CR/LF represent the carriage return line feed

ASCII Mode Special Command Characters	
Character	**Action at MicroLYNX**
<esc> Escape Key	Terminates all active operations and all running programs.
<^C> Ctrl+C Keys	Terminates all active operations and all running programs, forces a reset of the MicroLYNX.
<BKSP> Backspace Key	Moves the cursor back one in the buffer to correct a typing error.
<CR> or <LF> Carriage Return or Line Feed	Depending on the mode, either Single or Party. <LF> is not necessary in Single Mode communications. <CTRL+J> is the same as <LF> (0A Hex)

Table 5.7: ASCII Mode Special Command Characters

characters that are transmitted in response to the Enter key on the keyboard provided the ASCII setup specifies "Send line feeds with line ends". Note that there need not be a space between the data for the last parameter and the CR/LF. Also note that if there is only one parameter, the CR/LF would immediately follow the data for that parameter.

The command format in ASCII mode when the control module is in Party Mode (PARTY = TRUE) would be identical to that in Single Mode with the exception that the entire command would be preceded by the control module's address character (stored in DN) and terminated by a CTRL-J rather than ENTER:

<Address character><Mnemonic><white space><ASCII data for 1st parameter>, <ASCII data for 2nd parameter>, … , <ASCII data for nth parameter><CTRL-J>

Binary

Binary mode communications is faster than ASCII and would most likely be used in a system design where the communication speed is critical to system operation. This mode cannot be used with standard terminal software.

The command format in binary mode when the control module is in Single Mode (PARTY = FALSE) is:

<20H><character count><opcode><Field type for 1st parameter><IEEE hex data for 1st parameter><0EH><Field type for 2nd parameter><IEEE hex data for 2nd parameter><0EH> … <Field type for nth parameter><IEEE hex data for nth parameter><optional checksum>

Note that <20H> is 20 hex, the character count is the number of characters to follow the character count not including the checksum if one is being used. The OpCodes for control module instructions, variables, flags and keywords are given in Sections 15 and 16 of this document. The Field type byte will be one of the following based on the type of data that is expected for the specific parameter:

<0EH> is 0E hex, which is a separator character in this mode. Finally, the optional checksum will be included if CSE is TRUE and excluded if it is FALSE. If included, the checksum is the low eight bits of the complemented sixteen-bit sum of the address field (20H here), character count, OpCode, all data fields and separators (0E hex).

Binary Hex Codes	
Hex Code	**Data Type**
01	Label Text
02	ASCII Text
03	1 byte unsigned
04	2 byte signed
05	2 byte unsigned
06	4 byte signed
07	4 byte unsigned
08	4 byte float

Table 5.8: Binary Hex Codes

SECTION 6

Configuring the Digital I/O

Section Overview

This section covers the usage of the Isolated Digital and High Speed Differential I/O channels which are available on the LYNX System

- System I/O Availability by Module.
- The Isolated Digital I/O:
  - Configuring an Input
  - Setting the Digital Input Filtering for the Isolated I/O
  - Configuring an Output
  - Setting the Binary State of an I/O Group
- The Differential I/O:
  - The Clock Interface
  - Configuring an Input
  - Setting the Digital Input Filtering for the Differential I/O
  - Configuring an Output.
- Typical Functions of the Differential I/O.

System I/O Availability by Module

The LYNX System offers the designer ability to custom-tailor the LYNX System for their individual application needs. Below is a table illustrating the available configurations and the I/O set which whould be present with each configuration.

Allowable LYNX System I/O Configurations							
LYNX System	**LX-CM100**	**LX-CM200**	**LX-CM100 LX-DI100**	**LX-CM200 LX-DI100**	**LX-CM100 LX-DD100**	**LX-CM100 LX-DI100 LX-DD100**	**LX-CM100 LX-DC100**
GROUP 10 HIGH SPEED	11 & 12	11, 12, 13, 14 & 17	11 & 12	11, 12, 13, 14 & 17	11 - 18	11 - 18	11, 12, 13, 14 & 17
GROUP 20 ISOLATED	21 - 26	21 - 26	21 - 26	21 - 26	21 - 26	21 - 26	21 - 26
GROUP 30 ISOLATED	31 - 36	N/A	31 - 36	N/A	31 - 36	31 - 36	31 - 36
GROUP40 ISOLATED	N/A	N/A	41 - 46	41 - 46	N/A	41 - 46	41 - 46
GROUP 50 ISOLATED	N/A	N/A	51 - 56	51 - 56	N/A	51 - 56	N/A

Table 6.1: System I/O Availability by Module

The Isolated Digital I/O

The LYNX Control Module comes standard with a set of six (12) +5 to +24VDC I/O lines which may be programmed individually as either general purpose or dedicated inputs or outputs, or collectively as a group. The isolated digital I/O may also be expanded to twenty-four (24) lines in groups of six (6).

The I/O groups and lines are numbered in the following fashion:

Group 20 = Lines 21 - 26 (Standard)
Group 30 = Lines 31 - 36 (Standard)
Group 40 = Lines 41 - 46 (Optional)
Group 50 = Lines 51 - 56 (Optional)

The isolated digital I/O may be defined as either active HIGH or active LOW. When the I/O is configured as active HIGH, the level is +5 to +24 VDC and the state will be read as a "1". If the level is 0 VDC then the state will be read as "0". Inversely, If configured as active LOW, then the state of the I/O will be read as a "1" when the level is LOW, and a "0" when the level is HIGH. The active HIGH/LOW state is configured by the third parameter of the IOS variable, which is explained further on. The goal of this I/O configuration scheme is to maximize compatibility between the LYNX and standard sensors and switches.
The LYNX I/O scheme is a powerful tool for machine and process control. Because of this power, a level of complexity in setup and use is found that doesn't exist in controllers with a less capable I/O set.

Uses of the Isolated Digital I/O

The isolated I/O may be utilized to receive input from external devices such as sensors, switches or PLC outputs. When configured as outputs, devices such as relays, solenoids, LED's and PLC inputs may be controlled from the LYNX. Depending on the device connected, the input or output may be pulled-up to either the internal +5VDC supply or an external +5 to +24VDC supply, or the I/O lines may be pulled-down to ground.
These features, combined with the programmability and robust construction of the MicroLYNX I/O open an endless vista of possible uses for the I/O in your application.

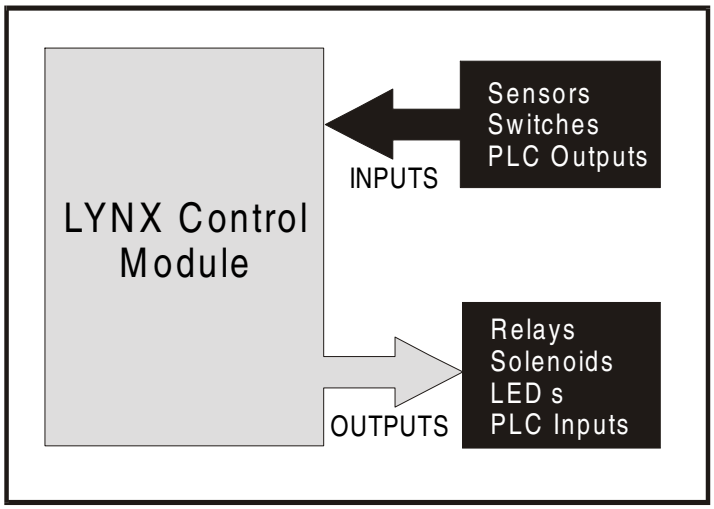


Figure 6.1: Isolated I/O Applications

Each I/O line may be individually programmed to any one of 8 dedicated input functions, 7 dedicated output functions, or as general purpose inputs or outputs. The I/O may be addressed individually, or as a group. The active state of the line or group may also be set. All of these possible functions are accomplished with of the IOS variable

The IOS Variable

The IOS variable has three parameters when used to configure the isolated digital I/O. These are:

1] **I/O Line Type:** Specifies the the type of I/O that the line or group will be configured as, i.e. general purpose or dedicated function.

2] **I/O Line Function:** Either an input or an output.

3] **Active State:** Specifies whether or not the line will be active HIGH or active LOW.

The default configuration of the standard I/O set is: 0,0,1. This means that by default each line in group 20 is configured to be a General Purpose (0), Input (0), which is active when HIGH (1). The following figure and exercises illustrate possible configurations of the IOS.

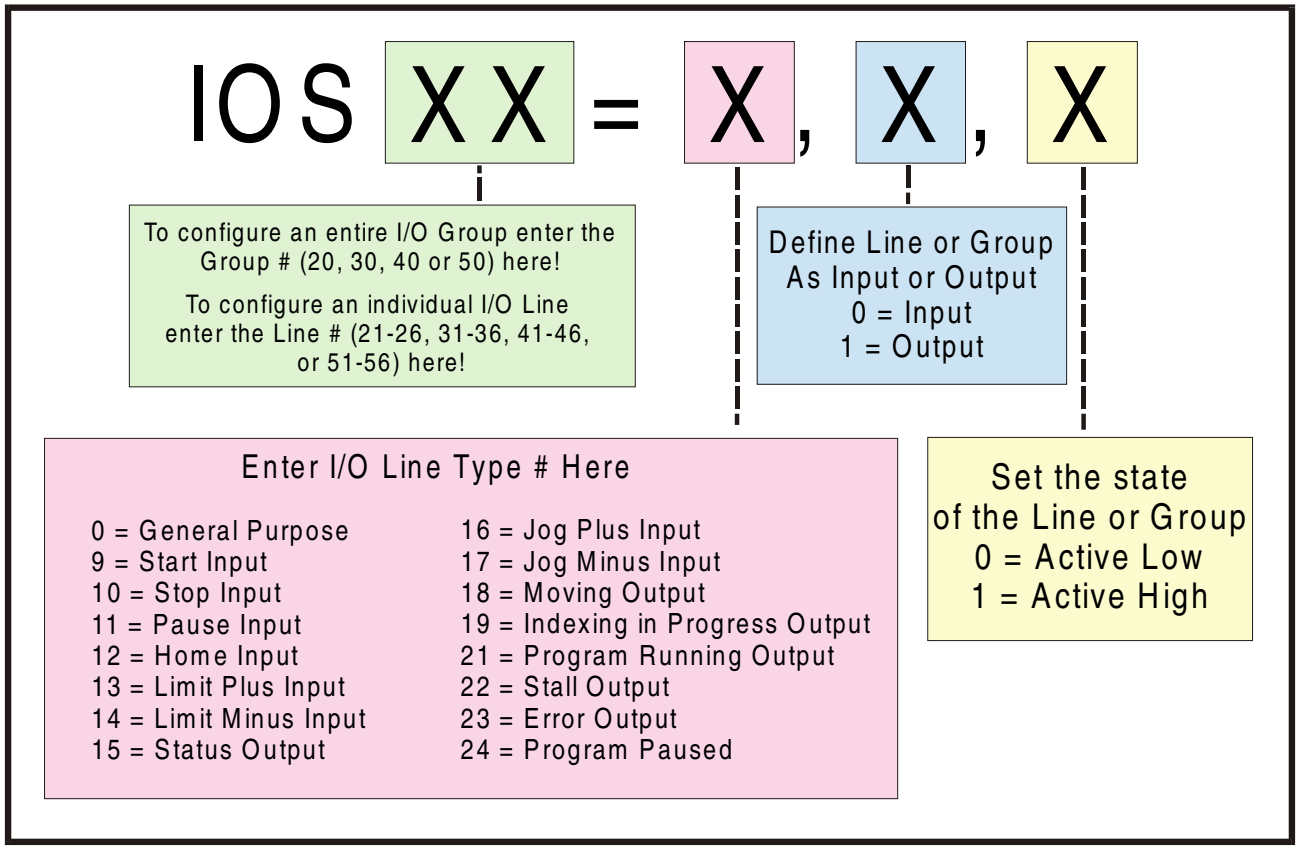


Table 6.2: IOS Variable Settings

NOTE: When configuring a dedicated input or output, the second parameter of the IOS Variable MUST match the function, either input or output, or an error will occur.

Configuring an Input

Figure 6.1 below illustrates the Input Equivalent Circuit of the Isolated I/O being used with a switch. To illustrate the usage of an input you will go through the steps to configure this switch to start a simple program at Line 1000 to index a motor 200 user units. First you must configure the I/O Line 21 as a "GO" input:

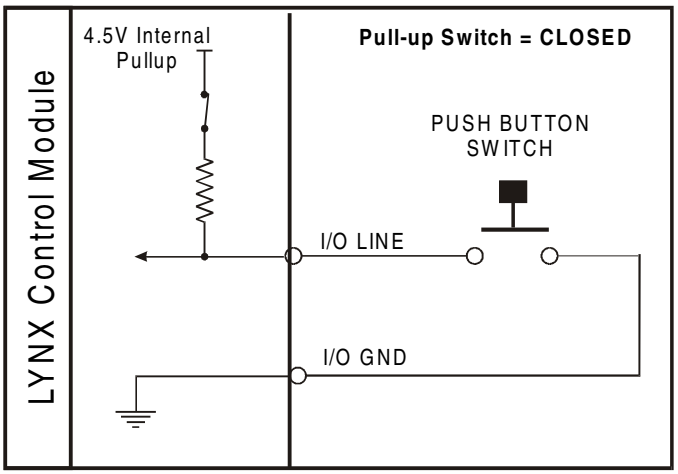


Figure 6.2: Isolated I/O Input

```
IOS 21 = 9, 0, 0
```

To break this command down:

IOS 21 - Identifies the I/O Line we are setting as 21.
9 - Configures the I/O Type to "GO".
0 - Configures I/O as Input.
0 - Configures I/O as Active LOW.

When the Input Type "GO" is selected it will always look to execute a program located at line 1 of program memory. Therefore, to run a program at line 1000 you must do the following:

```
PGM 1          'Records program at line 1 of memory space
EXEC 1000      'Execute program located at line 1000 of memory space
END            'Terminates Program
PGM            'Switches system back to immediate mode

PGM 1000       'Records program at line 1000 of memory space
MOVR 200       'Move relative to current position 200 user units
HOLD 2         'Hold program execution until specified motion is
               'completed
END
PGM
```

Configuring the Digital Filtering

User definable Digital filtering makes the LYNX well suited for noisy industrial environments. The filter setting is software selectable using the *IOF Variable* with a minimum guaranteed detectable pulse width of 18 microseconds to 2.3 milliseconds.

The table below illustrates the IOF settings.

The filter setting will reject any frequency above the specified bandwidth. For example:

```
IOF 2 = 3    'Set the Digital
Filter for I/O Group 20 to
3.44kHz
```

This setting will cause any signal above 3.44 kHz on I/O lines 21-26 to be rejected. The default filter setting for the isolated I/O groups is 7, or 215Hz.

IOF Filter Settings for the General Purpose Isolated I/O IOF=<num> (<num> = 0-7)		
Filter Setting	**Cutoff Frequency**	**Minimum Detectable Pulse Width**
0	27.5 kHz	18 microseconds
1	13.7 kHz	36 microseconds
2	6.89 kHz	73 microseconds
3	3.44 kHz	145 microseconds
4	1.72 kHz	290 microseconds
5	860 Hz	581 microseconds
6	430 Hz	1.162 milliseconds
7 (default)	215 Hz	2.323 milliseconds

Table 6.3: Digital Filter Settings for the Isolated I/O

Configuring an Output

Figure 6.2 following illustrates the Output equivalent circuit of the Isolated I/O. When used as an output the I/O line is able to sink 350mA continuous for each output, or a total of 1.5A for the entire I/O Group. See Section 9: *The Isolated I/O Module* for detailed specifications. In the usage example we will use an LED on I/O Line 31 for the load. We will use the same program from the input example, only we will use the output to light the LED while the motor is moving.

```
IOS 31 = 18, 1,  1
```

Using the table on page 27 we can break this setting down as follows:
IOS 31 - Identifies that I/O line 31 is being configured.
18 - Configures the I/O Type as "Moving".
1 - Configures the I/O line as an output.
1 - Configures the Line as "Active HIGH".

Now when the input program above is executed, the LED will be lit during the move.

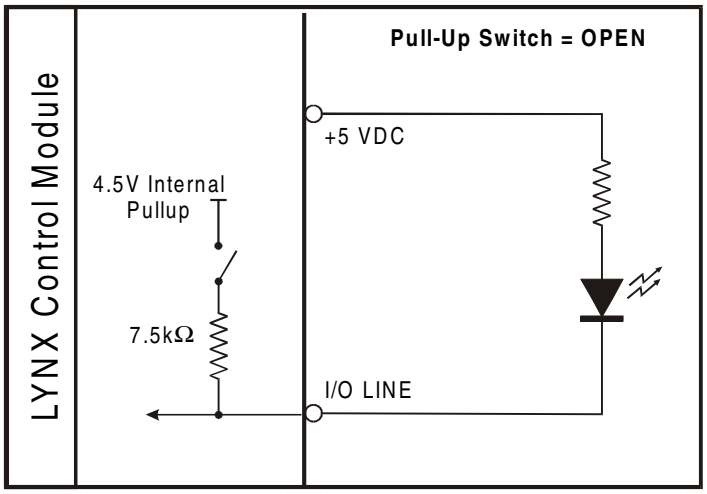


Figure 6.3: Isolated I/O Output

The IO Variable

After configuring the I/O by means of the IOS variable, we need to be able to do two things with the I/O.

1] Write to an output, or group of outputs, thus setting or changing its (their) state.
2] Read the states of either inputs or outputs. We can use this information to either display those states to our terminal, or to set up conditions for branches and subroutine calls within a program.

We can also use this command to write or read the state of an entire I/O group.

Read/Write a Single I/O Line

To read the state of a single input or output, the following would be typed into the terminal:

```
PRINT IO 21
```

The response from this would be 1 or 0, depending on the state of the line.
The state of an input or output in a program can be used to direct events within a LYNX program by either calling up a subroutine using the "CALL" instruction, or conditionally branching to another program address using the "BR" instruction. This would be done in the following fashion.

```
CALL MYSUB, IO 22=1
```

This would call up a subroutine labled "MYSUB" when I/O line 21 is active.

```
BR 200, IO 22=0
```

This would branch to address 200 when I/O line 22 is inactive.

Writing to an output is accomplished by entering the following into a terminal or program:

```
IO 21=1
IO 21=0
```

This would change the state of I/O line 21.

Read/Write an I/O Group

When using the IO variable to read the state of a group of inputs/outputs, or write to a group of outputs you would first want to configure the entire I/O group to be general purpose inputs or outputs using the IOS variable. In this case the response or input won't be a logic state of 1 or 0, but rather the decimal equivalent (0 to 63) of the 6 bit binary number represented by the entire group.

When addressing the I/O as a group the LSB (*Least Significant Bit*) will be line 1 of the group, (*e.g. 21, 31, 41, 51*). The MSB (*Most Significant Bit*) will be line 6 of the group (*e.g. 26, 36, 46, 56*).

The table on the left shows the bit weight of each I/O line in the group. It also illustrates the state should 6 LED's be connected to I/O group 20 when entering the IO variables in this exercise.

Configure the IOS variable such that group 20 is all general purpose outputs, active low or:

```
IOS 20 = 0,1,0
```

Enter the following in the terminal:

```
IO 20 = 35
```

As shown in the table I/O lines 26, 22 and 21 should be illuminated, 25, 24 and 23 should be off.

Enter this next:

```
IO 20 = 7
```

Now I/O 21, 22 and 23 should be illuminated.

```
IO 20 = 49
```

I/O 26, 25, and 21 are illuminated.

BIT WEIGHT DISTRIBUTION TABLE FOR GROUP 20 I/O					
I/O 26 MSB	I/O 25	I/O 24	I/O 23	I/O 22	I/O 21 LSB
32	16	8	4	2	1

BINARY STATE OF I/O GROUP 20 IO 20 = 35					
○	●	●	●	○	○
1	0	0	0	1	1
I/O 26 MSB	I/O 25	I/O 24	I/O 23	I/O 22	I/O 21 LSB

BINARY STATE OF I/O GROUP 20 IO 20 = 7					
●	●	●	○	○	○
0	0	0	1	1	1
I/O 26 MSB	I/O 25	I/O 24	I/O 23	I/O 22	I/O 21 LSB

BINARY STATE OF I/O GROUP 20 IO 20 = 49					
○	○	●	●	●	○
1	1	0	0	0	1
I/O 26 MSB	I/O 25	I/O 24	I/O 23	I/O 22	I/O 21 LSB

Table 6.4: Binary State of Outputs

NOTE: You can only write to General Purpose Outputs. If you attempt to write to and input or dedicated output type an error will occur!

The Differential I/O

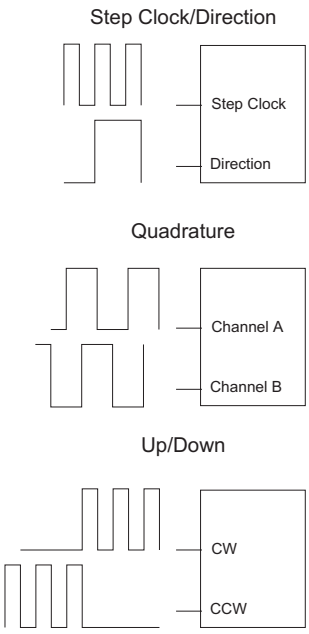


Figure 6.4: Clock Functions

The Clock Interface

The LYNX has four clock pairs that are used by the high speed I/O. One of these, clock pair 11 and 12, is fixed as an output and is used internally to provide step clock and direction pulses to Step Clock and Direction outputs located on Connector P1 of the LYNX Controller. The step clock output increments CTR1 (Counter 1). CTR1 may be read from or written to by software instructions in either program or immediate mode. The following table explains the clocks, as well as their default I/O line pair placement:

Clock Types Defined

There are three basic types of clocks that may be configured for the MicroLYNX, they are:

1] Quadrature
2] Step/Direction
3] Up/Down

These clock functions are illustrated in figure 6.3.

Quadrature

The quadrature clock function is the most commonly used input clock function. This is the default setting for each high speed I/O channel except 11 & 12. This clock function will typically be used for closed loop control (encoder feedback) or for following applications

Step/Direction

The step/direction clock funtion would typically be used in an application where a secondary or tertiary clock output is required to sequentially control an additional axis.

Up/Down

The up/down clock type would typically be used as an output function where a secondary axis is being driven by a stepper or servo drive with dual-clock direction control circuitry.

The Four Clocks				
Clock #	**I/O Line Pair**	**Slot Position**	**Counter**	**Function**
1	11 & 12	None	CTR1	This clock is internally generated motion clock. It provides step clock and directional control to the driver section. This clock is not available on any external connector.
2	13 & 14	Slot 2	CTR2	May be configured as an input or output. By default this is configured as a quadrature input. It can be configured as a secondary clock output electronically geared to CLK1.
3	15 & 16	Slot 3	CTR3	May be configured as an input or output. By default this is configured as a quadrature input. It can be configured as a tertiary clock output electronically geared to CLK1.
4	17	Slot 2	None	May be configured as a high speed input or an output. As an output it is a 1MHz reference clock.
	18	Slot 3	None	May be configured as a high seed input or output. As an output it is a 10MHz reference clock.

Table 6.5: The Four Clocks and Their Default Line Placement

Configuring The Differential I/O - The IOS Variable

The high speed differential I/O is configured by means of the IOS variable, and is used in the the same fashion in which the isolated I/O is configured. The main difference lies in that there are three additional parameters which need to be set in configuring the triggering, clock type, and ratio mode setting.

It is important to note that the high speed differential I/O lines may be used for the same input or output functions as the isolated digital I/O where the higher speed capabilities of the differential I/O is required. However, for purposes of this example we will only illustrate the clock functions associated with the high speed differential I/O. Figure 6.6 following illustrates the IOS variable settings for the high speed differential I/O.

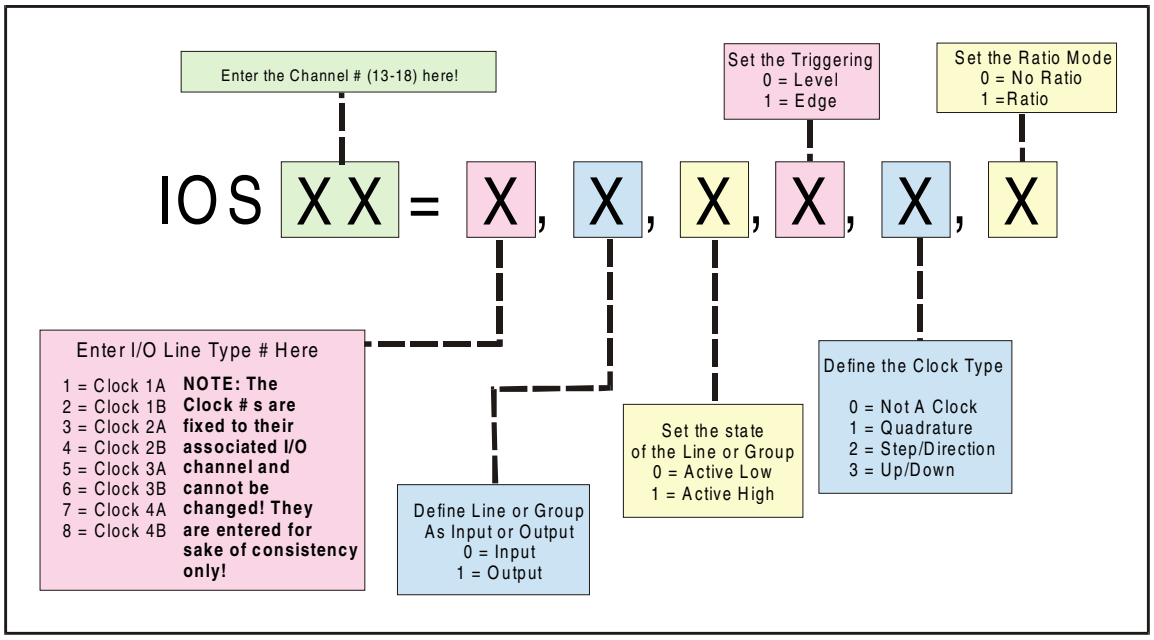


Figure 6.5: IOS Variable Settings for the High Speed Differential I/O

Configuring an Input

Clocks 2, 3 and 4 can be configured as high speed inputs, or as a general purpose input in the same fashion as the Isolated I/O. In configuring the Differential I/O line as a general purpose input you would typically use the "+" line of the line pair. You cannot use both lines as separate I/O lines. The figure below shows the Input Equivalent Circuit with the I/O line pair connected to channel A of a differential encoder. This feature

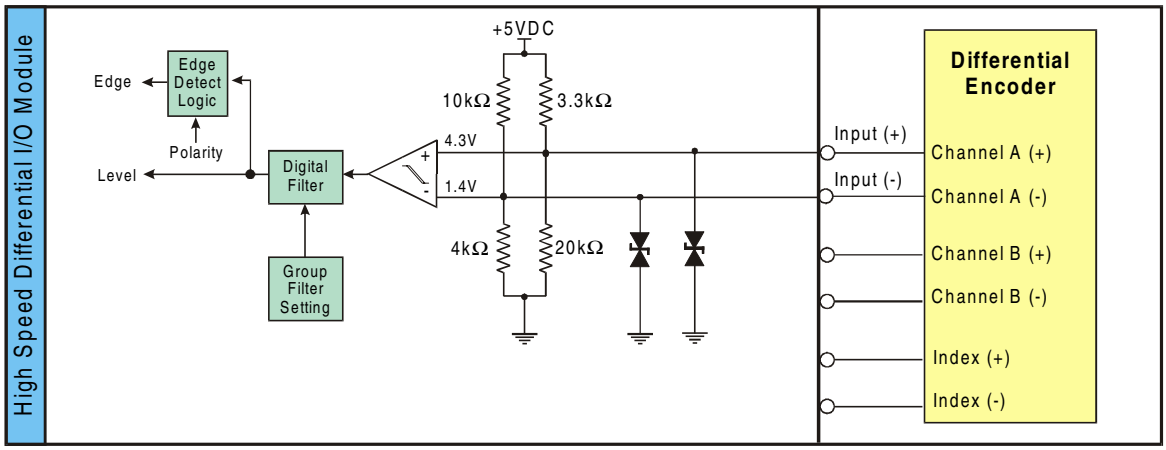


Figure 6.6: Differential I/O Input Equivalent Circuit

is demonstrated in *Typical Functions of the Differential I/O: Connecting and Using an Encoder.* Clocks 2, 3 and 4 are set up as Quadrature inputs by default. The defaults for each I/O Line Pair are:

IOS 13 = 3, 0, 1, 0, 1, 0
IOS 14 = 4, 0, 1, 0, 1, 0
IOS 15 = 5, 0, 1, 0, 1, 0
IOS 16 = 6, 0, 1, 0, 1, 0
IOS 17 = 7, 0, 1, 0, 1, 0
IOS 18 = 8, 0, 1, 0, 1, 0

Setting the Digital Input Filtering for the Differential I/O

User definable Digital filtering makes the LYNX well suited for noisy industrial environments. The filter setting is software selectable using the *IOF Variable* with a minimum guaranteed detectable pulse width of 18 microseconds to 2.3 milliseconds. The table below illustrates the IOF settings.

IOF Filter Settings for the High Speed Differential I/O IOF=<num> (<num> = 0-7)		
Filter Setting	**Cutoff Frequency**	**Minimum Detectable Pulse Width**
0 (default)	5.00 MHz	100 nanoseconds
1	2.50 MHz	200 nanoseconds
2	1.25 MHz	400 nanoseconds
3	625 kHz	800 nanoseconds
4	313 kHz	1.6 microseconds
5	156 kHz	3.2 microseconds
6	78.1 kHz	6.4 microseconds
7	39.1 kHz	12.8 microseconds

Table 6.6: Digital Filter Settings for the Differential I/O

Configuring an Output

The Differential I/O Group 10 has 3 Channels (Line Pairs 13 & 14, 15 & 16, and 17 & 18) that can be configured as an output by the user and One Channel (Line Pairs 11 & 12) that is configured as output only. (SCK and DIR on the Control Module.) These outputs can be configured as high speed outputs or 0 to 5VDC general purpose outputs by using the IOS variable. The high speed clock outputs have the following restrictions:

Line Pairs 11/12, 13/14 and 15/16 can be configured to Step Clock/Direction or Up/Down.

Line Pair 17/18 is limited to 1MHz Reference Out (17) and 10MHz Reference Out (18).

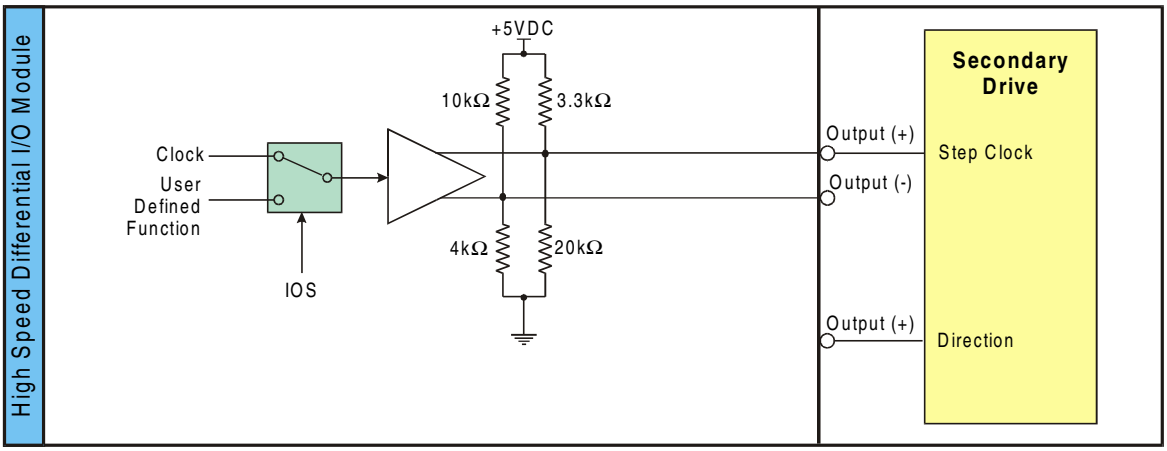


Figure 6.7: Differential I/O Output Equivalent Circuit

In the Equivalent Circuit in Figure 17 an Output is being used as Step or Direction on a driver.

For the configuration example, use I/O line 13 for the output. Since by default the line is a quadrature input we must configure it to be a Step/Direction Output by setting the IOS Variable to the following:
IOS 13 = 3, 1, 0, 1, 2, 0
This breaks down as:

IOS 13 - Identifies the line being configured as 13.
3 - Sets the I/O Type to Clock 2A (default).
1 - Sets it as an output.
0 - Sets Logic at Low True.
1 - Edge Triggered.
2 - Sets the Clock Type to Step/Direction.
0 - No Ratio.

Typical Functions of the Differential I/O

Connecting and Using an Encoder

The differential I/O module can be set up to receive encoder feedback using either a differential or a single ended output encoder. A differential output encoder would typically be connected to differential input pairs 13 and 14 (P1, pins 1 – 4) as the default setting for I/O 13 and 14 is set up to accept a quadrature encoder input. Channel A of the encoder would be connected to input pair 13 (P1, pins 1 & 2) and channel B would be connected to input pair 14 (P1, pins 3 & 4). A single ended output encoder would be connected to the positive inputs of the input pair. Whether you use a differential encoder or single ended encoder the same software commands and settings will be used.

In setting up your system to run with an encoder you will be using the following variables, flags, and instructions. The variables used with an encoder will be *MUNIT*, *EUNIT*, *CTR2*, and *POS*. The Encoder Enable Flag *EE*, and the instruction *MOVR* will be used. The block diagram to the left illustrates a LYNX system with the encoder and drive connections that will be used in this example.

The sequence of commands (in bold) used to make this setup function would be as follows:

```
'Set the MUNIT Variable to 51,200 steps/rev
MUNIT = 51200
'Set encoder enable to TRUE (1), default value = FALSE (0)
EE = 1
'Set the EUNIT (Encoder Units) variable to 800 (200 [Encoder Resolution] X 4 [Quadrature Input])
This means that 1 unit of motion, or 1 POS, is equal to 800 encoder counts. In this instance it will be
1 rotation of the motor.
EUNIT = 800
'Save the above flag and variable settings
SAVE
```

Now you may begin to use the motion command MOVR, as well as PRINT POS and PRINT CTR2 to see the number of encoder counts fed back to the system.

```
'Set the motor position to 0
POS = 0
'Move the motor 2 units (2 X EUNIT) relative to current position.
MOVR 2
```

'Print the value of CTR2. This value will indicate the number of encoder counts that the motor has moved. Your terminal should echo back the number "1600"

```
PRINT CTR2
'Print the position of the motor.  Your terminal should echo "2.000"
PRINT POS
```

By printing the variable CTR2 (CTR2 = EUNIT X POS) we can view the distance the motor has traveled in raw encoder counts, or, by printing POS you can see the distance of travel represented by number of units relative to 0.

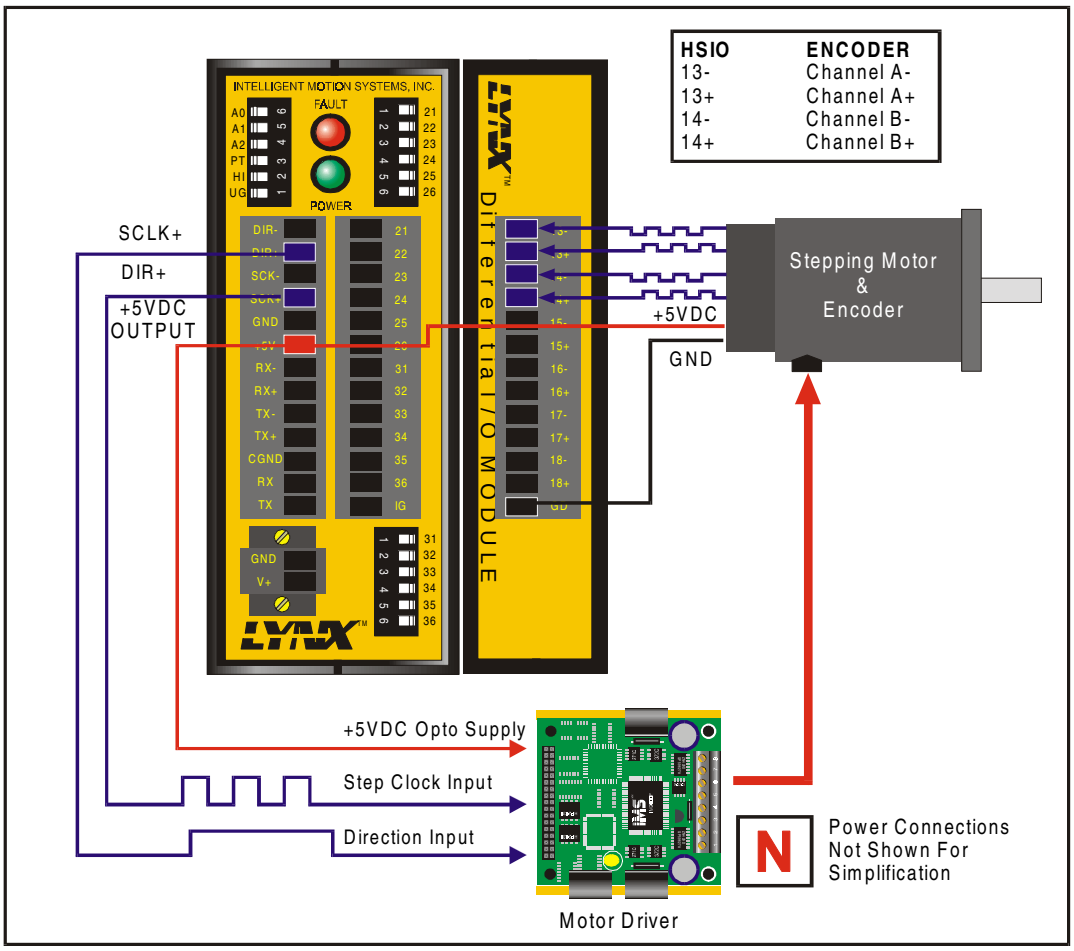


Figure 6.8: Connecting and Using an Encoder

Translating the EUNIT Variable to a Dimension of Distance

The EUNIT, or *Encoder Unit* variable, is the scaling factor used to translate Encoder steps to a dimension of distance, or, *user units*. At this point you should already be familiar with the MUNIT variable. The main difference between the two is as follows: By using MUNIT scaling factor you monitor the position of an axis based upon the value of CTR1, the register that contains the actual count of clock pulses sent to the drive. The number of pulses is then scaled to user units by setting the MUNIT Variable to the appropriate scaling factor for the type of units being used, be they inches, millimeters, degrees or etc. Then the POS variable tracks position in the user units specified. Example:

User Unit (POS) = CTR1 ÷ MUNIT where EE (Encoder Enable) Flag = FALSE (0)

By setting the state of EE, the master encoder function enable flag, to a true state you will monitor the position of an axis based upon the actual position of the motor shaft as it is fed back to the Control Module

by a motor mounted encoder. The actual count of encoder pulses received by the Control Module is maintained by the register CTR2, (if the encoder is connected to I/O line pair 13 &14) with the EUNIT variable scaling it to user units. Example:

User Unit (POS) = CTR2 ÷ EUNIT where EE (Encoder Enable) Flag = TRUE(1)

When using the EUNIT scaling factor it is important to understand that you MUST set the EUNIT variable AND the MUNIT variable to the same scaling factor for accurate position monitoring. In the example below you will use a hypothetical system designed from the following components:

An IMS IB462H Half/Full Step driver configured for Half Step Operation.
A 1.8° Stepping Motor mounted to a 20cm linear slide.
A 200 Line Encoder.

You will want to use millimeters for our user unit. The IB462H in half step mode will need 400 clock pulses to turn the motor one revolution. The pitch on the leadscrew is such that one millimeter of linear motion will require 25 clock pulses. 400 steps/rev ÷ 25 steps/mm = 16 mm/rev. Therefore, you would set the MUNIT variable as follows:

MUNIT = 400/16

Now, when you give a MOVR 20 instruction, the axis will index 20 millimeters. Now to set the EUNIT Variable. We have a 200 line encoder connected to a quadrature clock input. This will mean that 1 revolution will equal 800 Encoder Pulses, you will have to use the same scaling factor as we did for MUNIT as there will still be 16mm per revolution:

EUNIT = 800/16

Both values must be set, and both must be set to the same scaling factor. With the EE = 1 a MOVR 20 command will still index the axis 20 millimeters, but position will be maintained by CTR2.

Half Axis Operation (Follower)

In half axis mode the master clock is taken from a clock input 2, 3 or 4 (line pairs 13-14, 15-16 or 17-18) which have been set for input, clock type and ratio enabled. This is the factor at which the count rate out to the primary drive will follow the external clock in half axis mode. This clock input would typically be connected to differential input pairs 15 and 16 (P1, pins 5 – 8). This could be set up as any of the available clock types. If half axis mode is enabled (HAE), the primary axis of the control will follow the clock input with the ratio specified by the *HAS variable.*

In order to use the HAS (Half axis mode scaling) variable the *HAE flag* must be set to true (1). For example, to set the half axis scale factor to .5, where the drive will follow the external Clock input with a ratio of 1 count to the drive for every two counts from the external clock, you would use the command: SET HAS = .5 (or HAS = .5). Figure 6.7 illustrates the connections for using this mode of operation using a clock input from an encoder.

The sequence of commands used to make this setup function would be as follows:

```
'Set IOS 15 to ratio mode
IOS 15 = 5,0,1,0,1,1
'Set IOS 16 to ratio mode
IOS 16 = 6,0,1,0,1,1
'Half axis enable set to true
HAE = 1
'Half axis scaling to .5 (1 output clock pulse to every 2 input clock pulses)
HAS = .5
```

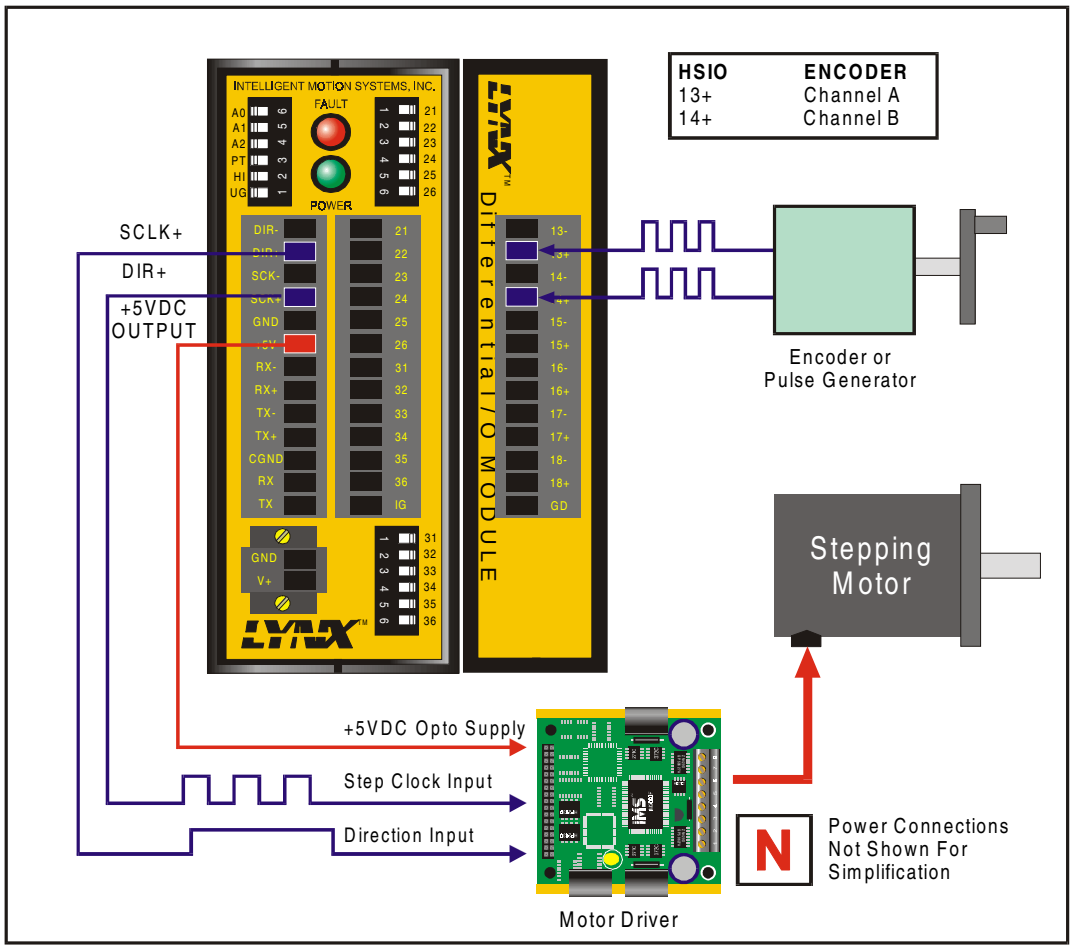


Figure 6.9 Half Axis Mode (Following)

NOTE: The HAS variable must be set to less than 1 or Error Code 9004, "Ratio Out of Range" will occur,

One and a Half Axis Operation (RATIOE)

A secondary drive can be connected to a pair of differential outputs. The secondary driver will operate off of the differential output pair 15 and 16 (I/O pair 13 and 14 can also operate in this mode). Setting the ratio mode to TRUE (1) for the differential output clock (IOS) specifies a secondary drive function. Then when ratio mode is enabled (*RATIOE*); the secondary axis will follow the primary axis with the ratio specified by the *RATIO* variable.

The sequence of commands used to make this setup function would be as follows:

```
'Set IOS 15 to step/direction clock          type, and ratio mode
IOS 15 = 5,0,1,0,2,1
'Set IOS 16 to step/direction clock          type, and ratio mode
IOS 16 = 6,0,1,0,2,1
'Set Ratio Mode Enable Flag to               TRUE (1)
```

```
RATIOE = 1
'Set RATIO variable to .5 for the                    secondary drive
RATIO = .5
```

With this setup, the motor on the secondary drive will move half the distance of the primary.

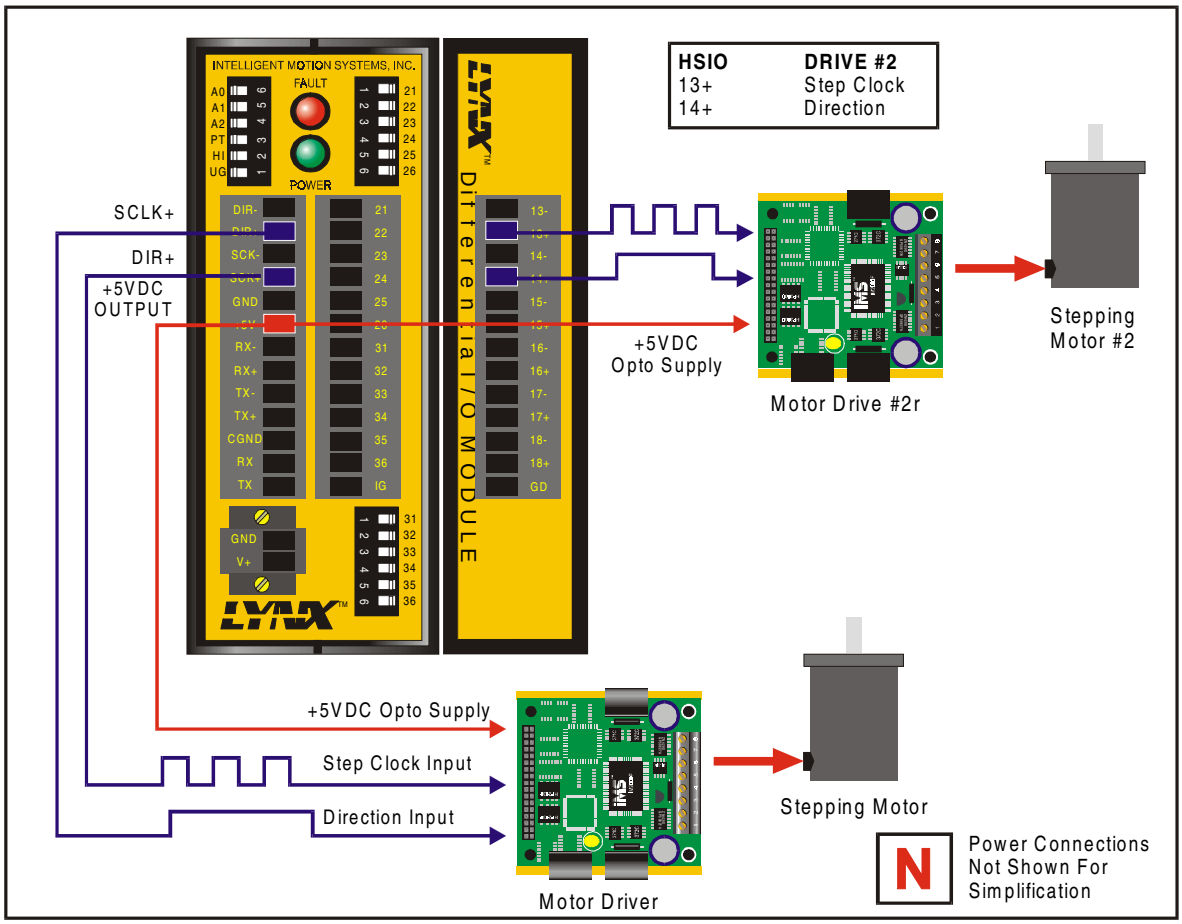


Figure 6.10: One and a Half Axis Operation

NOTE: The RATIO variable must be set to less than 2 or -2 or Error Code 9004, "Ratio Out of Range" will occur,

SECTION 7

The LYNX Control Module (LX-CM100-000)

Section Overview

This section will cover:

- Hardware Specifications
  - Environmental Specifications
  - Mechanical Specifications
  - Power Requirements
- Connection Overview
- LED Indicators
- Pin Assignments
- Switch Assignments

Hardware Specifications

Environmental Specifications

Operating Temperature ..... 0 to 50 degrees C
Storage Temperature ..... -20 to 70 degrees C
Humidity ..... 0 to 90% non-condensing

Mechanical Specification

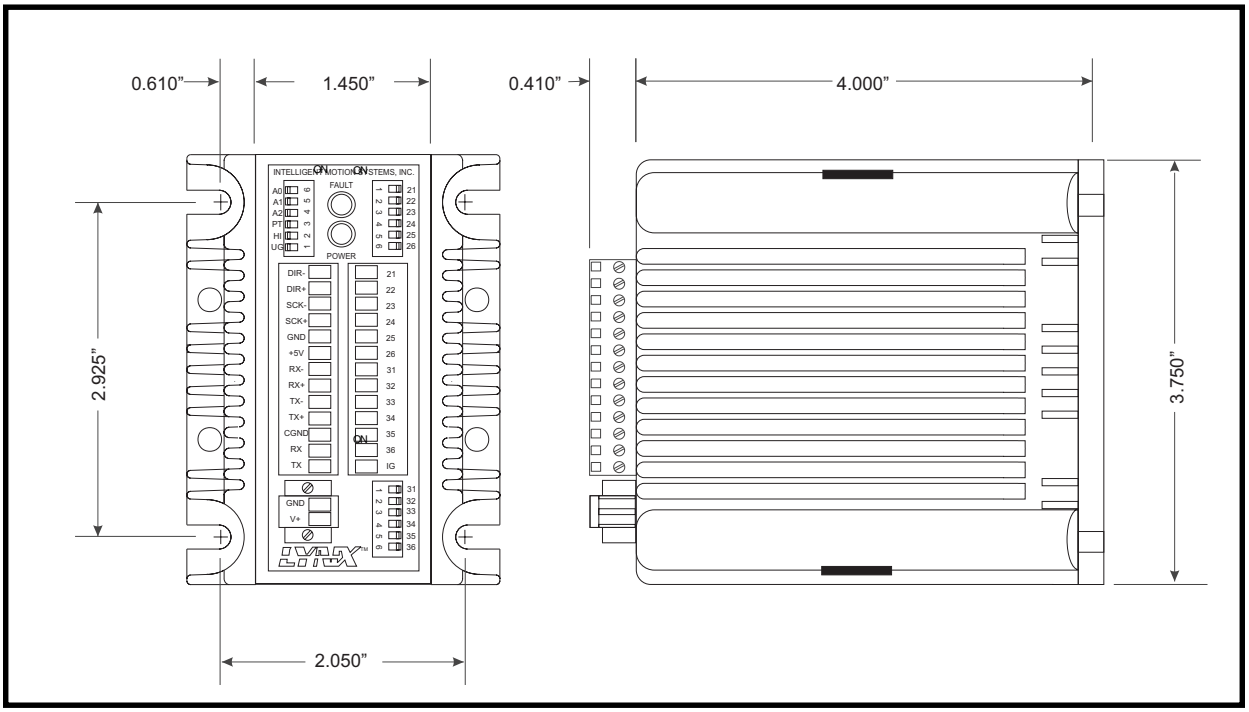


Figure 7.1: LYNX Control Module Dimensions

Power Requirements

Power Requirements and Specifications	
Input Voltage	+12 to +75 VDC Unregulated or +5VDC ±5%
Input Current	250mA (5VDC input) 165mA (+12VDC Input)* 95.0mA (+48 VDC Input)* 84.5mA (+75VDC Input)* *I/O and +5VDC output unloaded (Control Module Only)
Output Voltage	+5VDC ±5%
Output Current	150mA (Internally Limited

Table 7.1: Power Requirements for the LYNX Control Module

Connection Overview

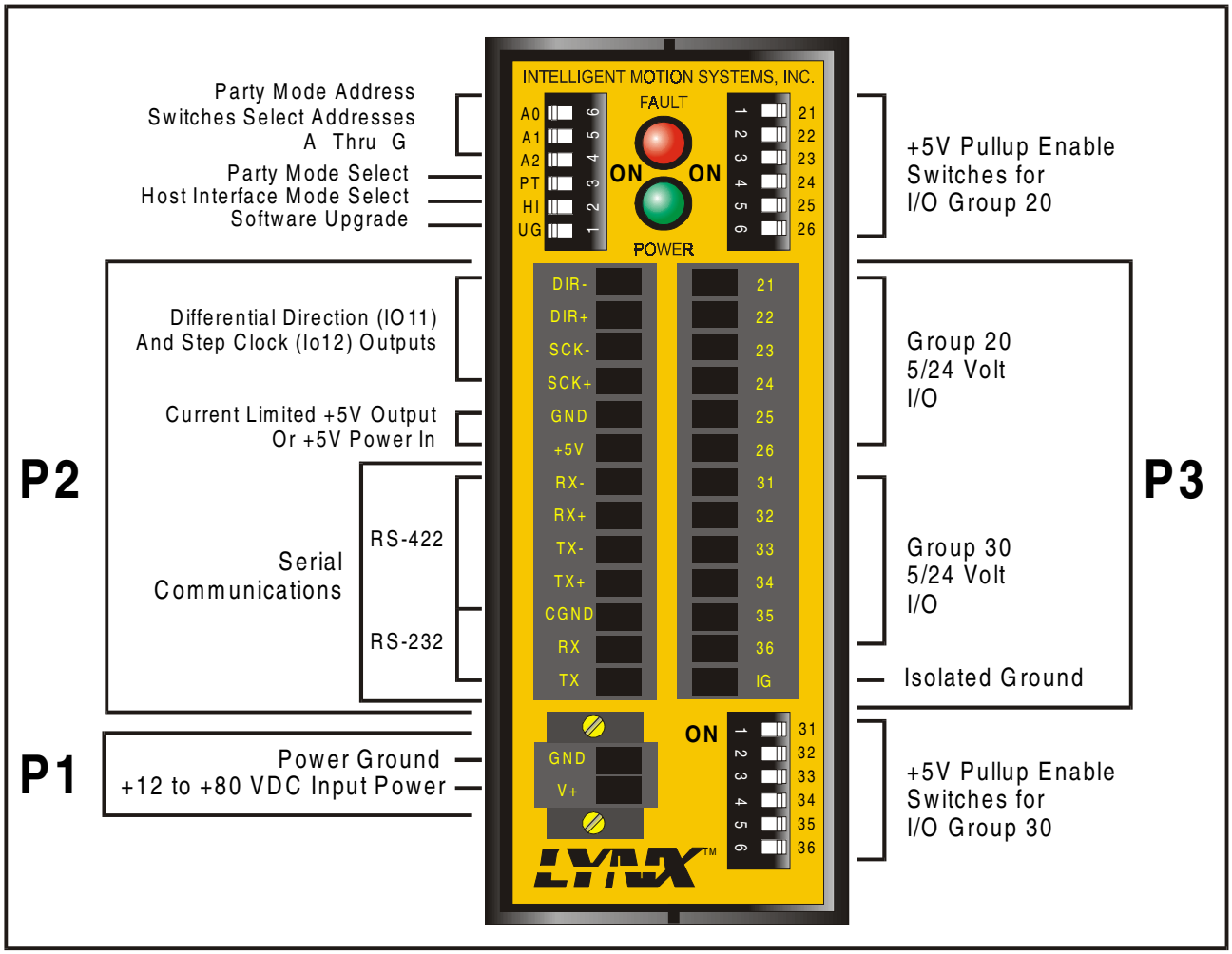


Figure 7.2: LYNX Control Module, Switches and Connections

LED Indicators

LED Color	Meaning
Green	Power On
Red	System or software fault detected. The user can choose to enable or disable the indicator by setting the FAULT flag. FAULT=TRUE (1) will cause the LED to illuminated whenever an ERROR occurs.

Table 7.2: LYNX Control Module LED Indicators

Pin Assignment and Description

P1 - Two Position Screw Lock Terminal: Input Power Connection		
Pin #	**Function**	**Description**
1	Power Ground	Power ground for the unregulated power supply.
2	Unregulated Power Supply Input (V+)	12 – 75 VDC unregulated power input if an external power supply is to be used.

Table 7.3: LYNX Control Module Connector P1 Pin Configuration

P2 - 13 Position Removeable Terminal Connector: Motion Signals, Regulated Power and Communications		
Pin #	**Function**	**Description**
1	Direction - (I/O 11)	Pins 1 and 2 are the differentially buffered signal for group 1, #1 or I/O 11. The default for this signal is the direction output for the primary motor drive of the controller. If desired, this signal may be programmed as a quadrature or up/down clock type or a user output. This I/O may not be programmed as an input.
2	Direction + (I/O 11)	See description above.
3	Step Clock - (I/O 12)	Pins 3 and 4 are the differentially buffered signal for group 1, #2 or I/O 12. The default for this signal is the step clock output for the primary motor drive of the controller. If desired, this signal may be programmed as a quadrature or up/down clock type or a user output. This I/O may not be programmed as an input.
4	Step Clock + (I/O 12)	See description above.
5	Ground (GND)	Common to the power ground on pin 1 of connector P1. This is provided as a signal return for the motion control signals and the power return for the 5VDC in/out on pin 6.
6	+5VDC	This can be 5 volts in or out. When the control module is powered via connector P1, this terminal provides up to 150 ma of regulated 5VDC for user circuits such as encoders. If desired, however, this terminal may be used as a power input connection. It should be noted that a fully configured LYNX system may require up to 800 ma current from this 5 VDC supply.
7	RS-485 RX- Input	Pins 7 and 8 are the differential receive inputs for the RS-485 communications interface. They should be left disconnected if they are not used. For specific connection information, see Section 5: The Communications Interface.
8	RS-485 RX+ Input	See description above.
9	RS-485 TX- Output	Pins 9 and 10 are the differential transmit outputs for the RS-485 communications interface. They should be left disconnected if they are not used. For specific connection information, see Section 5: The Communications Interface.
10	RS-485 TX+ Output	See description above.
11	Communications Ground (CGND)	Isolated communications ground signal for both RS-485 and RS-232. For specific connection information, see Section 5: The Communications Interface.
12	RS-232 RX Input	Receive input from the host computer. For specific connection information, see Section 5: The Communications Interface.
13	RS-232 TX Output	Transmit output to the host computer. For specific connection information, see Section 5: The Communications Interface.

Table 7.4: LYNX Control Module Connector P2 Pin Configuration

P3 - 13 Position Removeable Terminal Connector: Isolated Digital I/O		
Pin #	**Function**	**Description**
1 - 8	I/O Group 20 Lines 21 - 26	Signals are individually programmable as inputs or outputs (see description of the IOS command in the Part 3: Software Reference of this manual). Inputs are CMOS logic level compatible and can accept inputs to 28 volts. Noise rejection is available via digital filtering. Outputs are open drain. I/Os each have individually switchable 7.5 Kohm pull up resistors to 5VDC. Outputs can switch inductive, resistive or incandescent loads. Refer to Section 6: Configuring the Digital I/O for usage and specifications.
7 - 12	I/O Group 30 Lines 31 - 36	See description above.
13	Isolated I/O Ground	Isolated common signal return for groups 20 and 30 I/O. Isolated from the power and communication grounds.

Table 7.5: LYNX Control Module Connector P3 Pin Configuration

Switch Assignments

Configuration Switches: Read at Power-On or System Reset. Mat be Overidden by Software Settings		
Switch #	**Function**	**Description**
1	Firmware Upgrade	When this switch is on, the controller firmware may be upgraded using the IMS upgrade program.
2	Host Interface	When this switch is on, the controller will act as the Host Interface Controller for communications in a multiple controller system. When it is off, the controller is a slave in the system and will not act as the host interface. For more information, see Section 5: The Communications Interface. This switch may be overridden in software by the HOST flag.
3	Party Mode	When this switch is on, party mode communications is selected. When it is off, single mode communications is selected. For more information, see Section 5: The Communications Interface.
4 5 6	Party Mode Address Bit 0 - A0 Party Mode Address Bit 1 - A1 Party Mode Address Bit 2 - A2	Sets party mode communications node address. See also DN instruction in the Software Reference A2 A1 A0 Address OFF OFF OFF None OFF OFF ON "A" OFF ON OFF "B" OFF ON ON "C" ON OFF OFF "D" ON OFF ON "E" ON ON OFF "F" ON ON ON "G"

Table 7.6: LYNX Control Module Configuration Switches

Group 20 I/O Pull-Up Switches: Can Be Changed at any Time, Usable for Exercising Inputs		
Switch #	**Function**	**Description**
1 - 6	Individual Switches for I/O Group 20 Pull-Ups.	When this switch is on, the I/O is pulled up through an internal 7.5 Kohm resistor to 5VDC. Can be used to simulate the activation of an input while testing system software..

Table 7.7: LYNX Control Module Group 20 I/O Pull-up Switches

Group 30 I/O Pull-Up Switches: Can Be Changed at any Time, Usable for Exercising Inputs		
Switch #	**Function**	**Description**
1 - 6	Individual Switches for I/O Group 30 Pull-Ups.	When this switch is on, the I/O is pulled up through an internal 7.5 Kohm resistor to 5VDC. Can be used to simulate the activation of an input while testing system software..

Table 7.8: LYNX Control Module Group 30 I/O Pull-up Switches

SECTION 8

The LYNX Control Module (Combination)

Section Overview

The Control Module (Combination) (IMS Part # LX-CM200-000) offers the user of purchasing a LYNX Control Module with 3 differential I/O Channels and 6 Isolated I/O lines instead of the standard 2 Isolated I/O groups. This section will cover:

- Hardware Specifications
  - Environmental Specifications
  - Mechanical Specifications
  - Power Requirements
- Connection Overview
- LED Indicators
- Pin Assignments
- Switch Assignments

Hardware Specifications

Environmental Specifications

Operating Temperature ..... 0 to 50 degrees C
Storage Temperature ..... -20 to 70 degrees C
Humidity ..... 0 to 90% non-condensing

Mechanical Specification

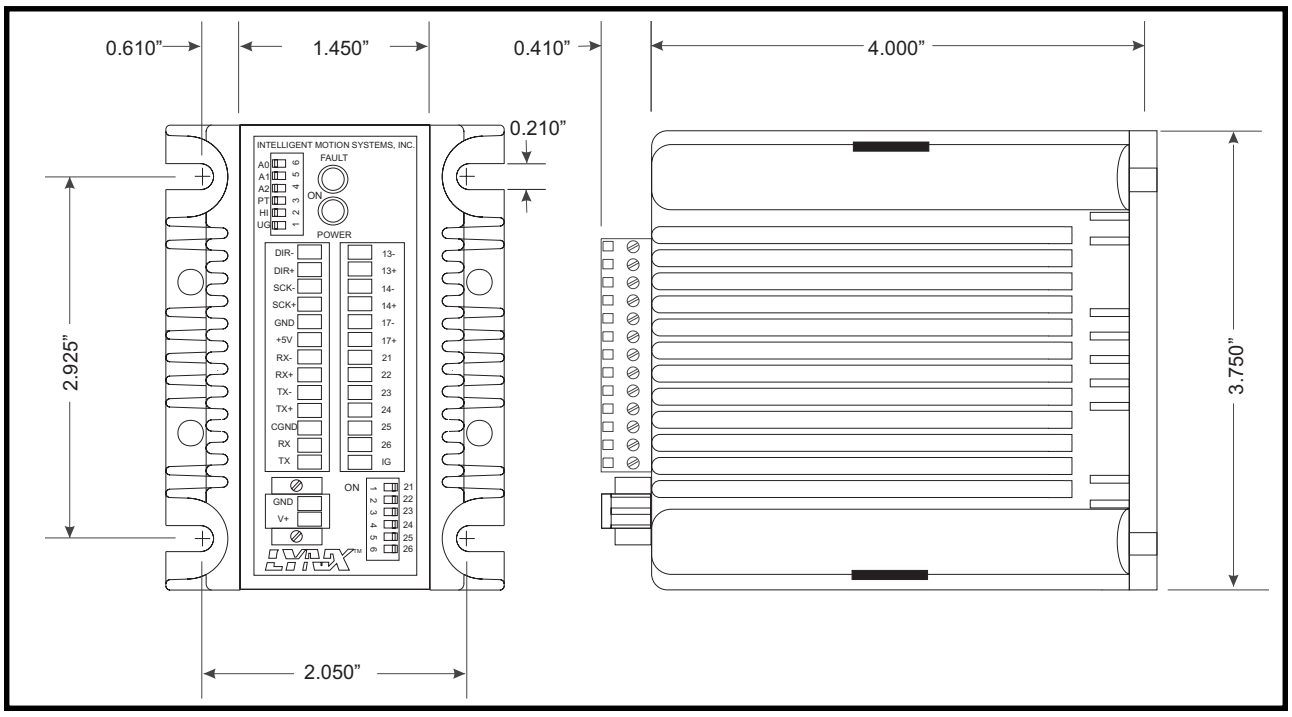


Figure 8.1: LYNX Control Module (Combination) Dimensions

Power Requirements

Power Requirements and Specifications	
Input Voltage	+12 to +75 VDC Unregulated or +5VDC ±5%
Input Current	250mA (5VDC input) 165mA (+12VDC Input)* 95.0mA (+48 VDC Input)* 84.5mA (+75VDC Input)* *I/O and +5VDC output unloaded (Control Module Only)
Output Voltage	+5VDC ±5%
Output Current	150mA (Internally Limited

Table 8.1: Power Requirements for the LYNX Control Module (Combination)

Connection Overview

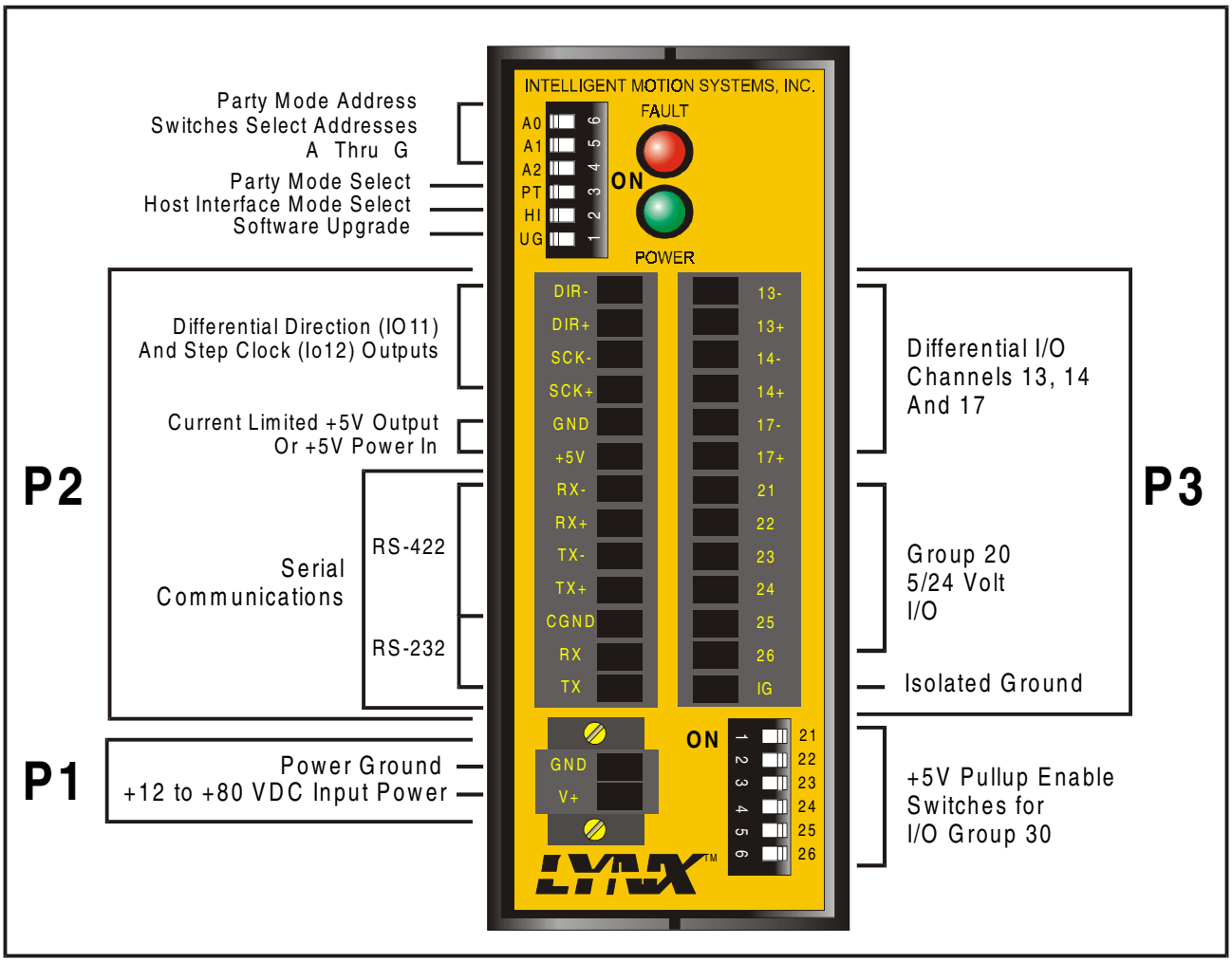


Figure 8.2: LYNX Control Module (Combination) Connections and Switches

LED Indicators

LED Color	Meaning
Green	Power On
Red	System or software fault detected. The user can choose to enable or disable the indicator by setting the FAULT flag. FAULT=TRUE (1) will cause the LED to illuminated whenever an ERROR occurs.

Table 8.2: LYNX Control Module LED Indicators

Pin Assignment and Description

P1 - Two Position Screw Lock Terminal: Input Power Connection		
Pin #	**Function**	**Description**
1	Power Ground	Power ground for the unregulated power supply.
2	Unregulated Power Supply Input (V+)	12 – 75 VDC unregulated power input if an external power supply is to be used.

Table 8.3: LYNX Combination Control Module Connector P1 Pin Configuration

P2 - 13 Position Removeable Terminal Connector: Motion Signals, Regulated Power and Communications		
Pin #	**Function**	**Description**
1	Direction - (I/O 11)	Pins 1 and 2 are the differentially buffered signal for group 1, #1 or I/O 11. The default for this signal is the direction output for the primary motor drive of the controller. If desired, this signal may be programmed as a quadrature or up/down clock type or a user output. This I/O may not be programmed as an input.
2	Direction + (I/O 11)	See description above.
3	Step Clock - (I/O 12)	Pins 3 and 4 are the differentially buffered signal for group 1, #2 or I/O 12. The default for this signal is the step clock output for the primary motor drive of the controller. If desired, this signal may be programmed as a quadrature or up/down clock type or a user output. This I/O may not be programmed as an input.
4	Step Clock + (I/O 12)	See description above.
5	Ground (GND)	Common to the power ground on pin 1 of connector P1. This is provided as a signal return for the motion control signals and the power return for the 5VDC in/out on pin 6.
6	+5VDC	This can be 5 volts in or out. When the control module is powered via connector P1, this terminal provides up to 150 ma of regulated 5VDC for user circuits such as encoders. If desired, however, this terminal may be used as a power input connection. It should be noted that a fully configured LYNX system may require up to 800 ma current from this 5 VDC supply.
7	RS-485 RX- Input	Pins 7 and 8 are the differential receive inputs for the RS-485 communications interface. They should be left disconnected if they are not used. For specific connection information, see Section 5: The Communications Interface.
8	RS-485 RX+ Input	See description above.
9	RS-485 TX- Output	Pins 9 and 10 are the differential transmit outputs for the RS-485 communications interface. They should be left disconnected if they are not used. For specific connection information, see Section 5: The Communications Interface.
10	RS-485 TX+ Output	See description above.
11	Communications Ground (CGND)	Isolated communications ground signal for both RS-485 and RS-232. For specific connection information, see Section 5: The Communications Interface.
12	RS-232 RX Input	Receive input from the host computer. For specific connection information, see Section 5: The Communications Interface.
13	RS-232 TX Output	Transmit output to the host computer. For specific connection information, see Section 5: The Communications Interface.

Table 8.4: LYNX Combination Control Module Connector P2 Pin Configuration

P3 - 13 Position Removeable Terminal Connector: Combination I/O		
Pin #	**Function**	**Description**
1	I/O 13-	Pins 1 and 2 are the differentially buffered signal for group 10, I/O 13.This channel is configured by means of the IOS Instruction. For usage details see Section 6: Configuring the Digital I/O.
2	I/O 13+	See description above.
3	I/O 14-	Pins 3 and 4 are the differentially buffered signal for group 10, I/O 14.This channel is configured by means of the IOS Instruction. For usage details see Section 6: Configuring the Digital I/O.
4	I/O 14+	See description above.
5	I/O 17-	Pins 5 and 6 are the differentially buffered signal for group 10, I/O 17 .This channel is configured by means of the IOS Instruction. For usage details see Section 6: Configuring the Digital I/O.
6	I/O 17+	See description above.
7 - 12	I/O Group 20 Lines 21 - 26	Signals are individually programmable as inputs or outputs (see description of IOS command in Part 3: The Software Reference of this manual). Inputs are CMOS logic level compatible and can accept inputs to 24 volts. Noise rejection is available via digital filtering. Outputs are open drain. I/Os each have individually switchable 7.5 Kohm pull up resistors to 5VDC. Outputs can switch inductive, resistive or incandescent loads. Refer to Section 6: Configuring the Digital I/O for more information.
13	Isolated Ground For Group 20 I/O	solated common signal return for group 20 I/O. Isolated from the power and communication grounds.

Table 8.5: LYNX Combination Control Module Connector P3 Pin Configuration

Switch Assignments

Configuration Switches: Read at Power-On or System Reset. Mat be Overidden by Software Settings		
Switch #	**Function**	**Description**
1	Firmware Upgrade	When this switch is on, the controller firmware may be upgraded using the IMS upgrade program.
2	Host Interface	When this switch is on, the controller will act as the Host Interface Controller for communications in a multiple controller system. When it is off, the controller is a slave in the system and will not act as the host interface. For more information, see Section 5: The Communications Interface. This switch may be overridden in software by the HOST flag.
3	Party Mode	When this switch is on, party mode communications is selected. When it is off, single mode communications is selected. For more information, see Section 5: The Communications Interface.
4	Party Mode Address Bit 0 - A0	Sets party mode communications node address. See also DN instruction in the Software Reference A2 A1 A0 Address OFF OFF OFF None OFF OFF ON "A" OFF ON OFF "B" OFF ON ON "C" ON OFF OFF "D" ON OFF ON "E" ON ON OFF "F" ON ON ON "G"
5	Party Mode Address Bit 1 - A1	
6	Party Mode Address Bit 2 - A2	

Table 8.6: LYNX Combination Control Module Configuration Switches

Group 20 I/O Pull-Up Switches: Can Be Changed at any Time, Usable for Exercising Inputs		
Switch #	**Function**	**Description**
1 - 6	Individual Switches for I/O Group 20 Pull-Ups.	When this switch is on, the I/O is pulled up through an internal 7.5 Kohm resistor to 5VDC. Can be used to simulate the activation of an input while testing system software..

Table 8.7: LYNX Combination Control Module Group 20 I/O Pull-up Switches

SECTION 9

The Isolated Digital I/O Module

Section Overview

The Isolated I/O Module (IMS Part # LX-DI100-000) offers the user of adding an addition 12 Isolated 5 to 24VDC General Purpose I/O lines in two groups of six each (Groups 40 and 50) for a total of 24 individually programmable/O when used with the LYNX Control Module. (18 When used with the Control Module (Combination)).

- Hardware Specifications
  - Environmental Specifications
  - Mechanical Specifications
- Pin Assignments
- Switch Assignments
- Input Specifications
- Input Filtering
- Output Specifications

Hardware Specifications

Environmental Specification

Operating Temperature ..... 0 to 50 degrees C
Storage Temperature ..... -20 to 70 degrees C
Humidity ..... 0 to 90% non-condensing

Mechanical Specification

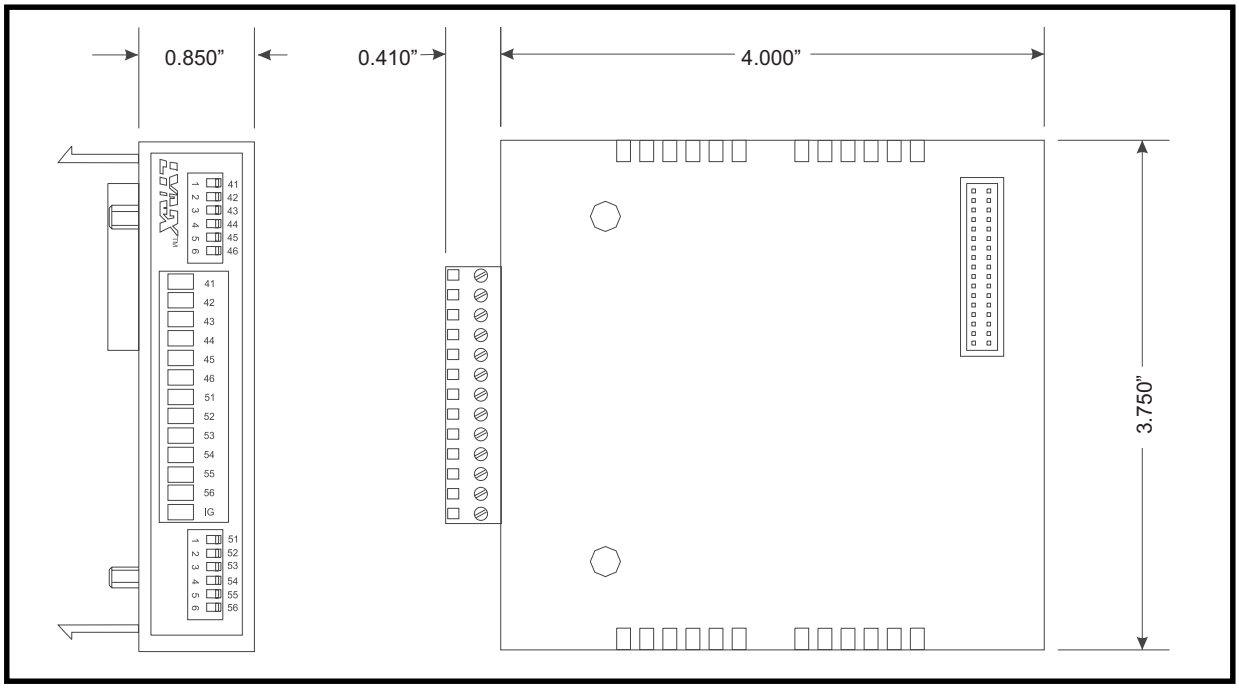


Figure 9.1: LYNX Isolated I/O Module Dimensions

Connection Overview

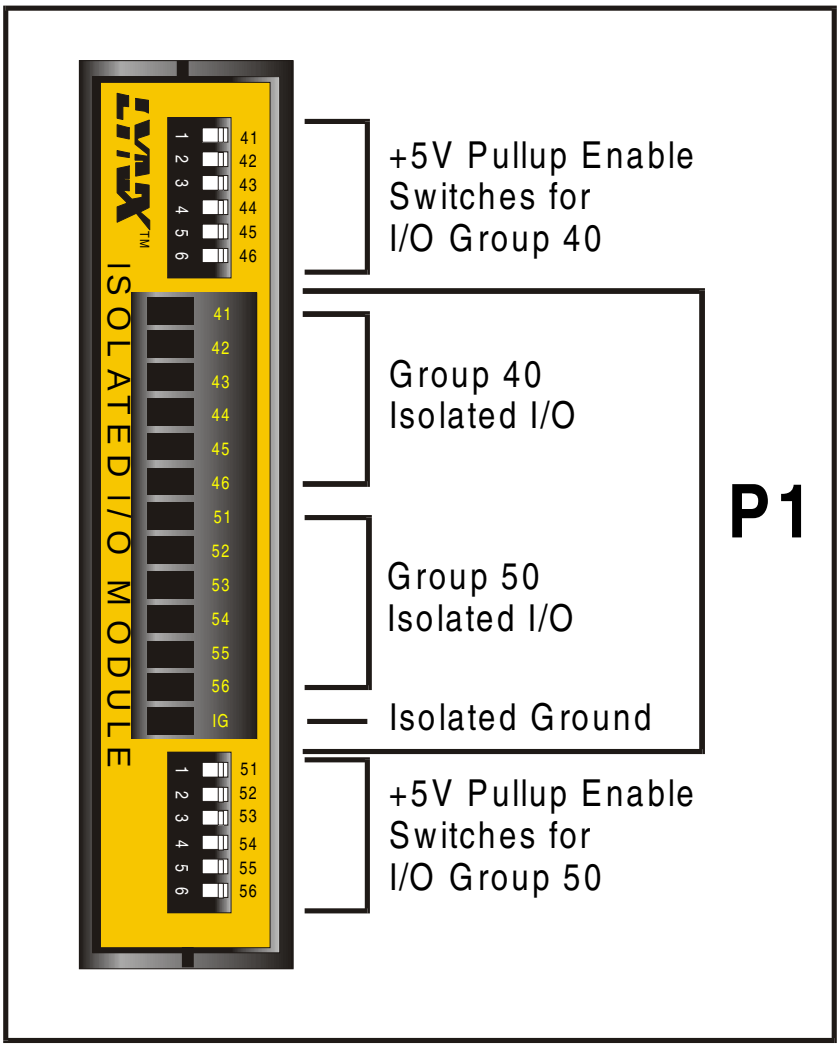


Figure 9.2: Isolated Digital I/O Module Connection Overview

Pin Assignments And Description

P1 - 13 Position Removeable Terminal Connector: Isolated Digital I/O		
Pin #	**Function**	**Description**
1 - 8	I/O Group 40 Lines 41 - 46	Signals are individually programmable as inputs or outputs (see description of the IOS command in the Part 3: Software Reference of this manual). Inputs are CMOS logic level compatible and can accept inputs to 24 volts. Noise rejection is available via digital filtering. Outputs are open drain. I/Os each have individually switchable 7.5 Kohm pull up resistors to 5VDC. Outputs can switch inductive, resistive or incandescent loads. Refer to Section 6: Configuring the Digital I/O for usage and specifications.
7 - 12	I/O Group 50 Lines 51 - 56	See description above.
13	Isolated I/O Ground	Isolated common signal return for groups 40 and 50 I/O. Isolated from the power and communication grounds.

Table 9.1: Isolated Digital I/O Module P1 Connector Pin Configuration

Switch Assignments And Description

Group 40 I/O Pull-Up Switches: Can Be Changed at any Time, Usable for Exercising Inputs		
Switch #	**Function**	**Description**
1 - 6	Individual Switches for I/O Group 40 Pull-Ups.	When this switch is on, the I/O is pulled up through an internal 7.5 Kohm resistor to 5VDC. Can be used to simulate the activation of an input while testing system software..

Table 9.2: Isolated I/O Module Group 40 I/O Pull-up Switches

Group 50 I/O Pull-Up Switches: Can Be Changed at any Time, Usable for Exercising Inputs		
Switch #	**Function**	**Description**
1 - 6	Individual Switches for I/O Group 50 Pull-Ups.	When this switch is on, the I/O is pulled up through an internal 7.5 Kohm resistor to 5VDC. Can be used to simulate the activation of an input while testing system software..

Table 9.3: Isolated I/O Module Group 50 I/O Pull-up Switches

Input Specifications

Isolated I/O Input Specifications	
Voltage Range	0 to 28 VDC (Transient Protected to 60 Volts)
Low Level	< 1.5V
High Level	> 3.5V
Open Circuit Input Voltage	Pull-Up Switch ON = 4.5V Pull-Up Switch OFF = 0V

Table 9.4: Isolated I/O Module Input Specifications

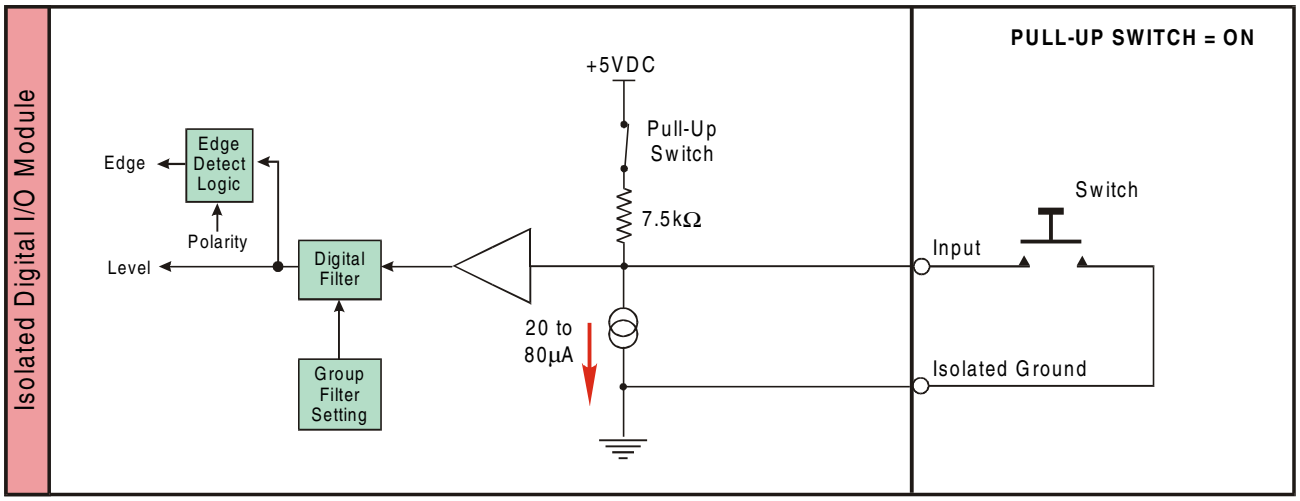


Figure 9.3: LYNX Isolated I/O Input Equivalent Circuit

Input Filtering

User definable Digital filtering makes the LYNX well suited for noisy industrial environments. The filter setting is software selectable using the *IOF Variable* with a minimum guaranteed detectable pulse width of 18 microseconds to 2.3 milliseconds.

The table at right illustrates the IOF settings.

IOF Filter Settings for the General Purpose Isolated I/O IOF=<num> (<num> = 0-7)		
Filter Setting	**Cutoff Frequency**	**Minimum Detectable Pulse Width**
0	27.5 kHz	18 microseconds
1	13.7 kHz	36 microseconds
2	6.89 kHz	73 microseconds
3	3.44 kHz	145 microseconds
4	1.72 kHz	290 microseconds
5	860 Hz	581 microseconds
6	430 Hz	1.162 milliseconds
7 (default)	215 Hz	2.323 milliseconds

Table 9.5: Digital Filter Settings for the Isolated I/O

Output Specifications

Isolated I/O Input Specifications	
Load Supply Voltage	28 VDC Maximum
FET On Resistance	2Ω Maximum (Tj = 125°C)
Continuous Sink Current	350mA Maximum Each Output (Ta = 25°C)
Maximum Group Sink	1.5A (Thermally Limited)
Open Circuit Output Voltage	Pull-up Switch ON = 4.5V Pull-up Switch OFF = 0V

Table 9.6: Digital Filter Settings for the Isolated I/O

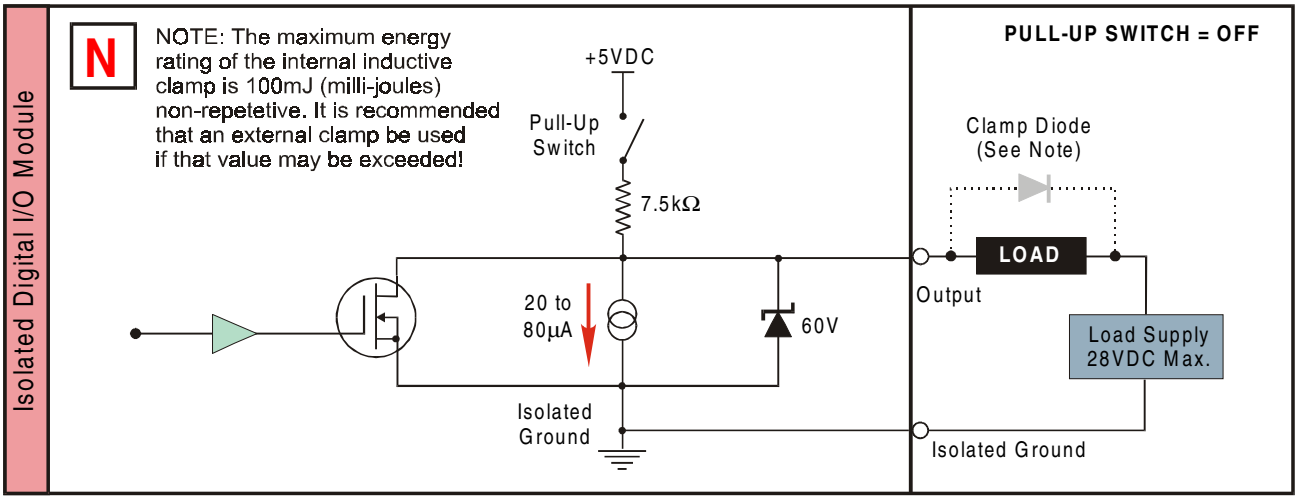


Figure 9.4: LYNX Isolated I/O Output Equivalent Circuit

SECTION 10

The Differential Digital I/O Module

Section Overview

A LYNX system may contain an optional Differential I/O Module which provides six (6) high speed differential I/Os. These I/Os can be used as clock inputs or outputs or general purpose I/O. Along with the differential motion I/Os (P1, pins 1 – 4) of the LYNX Control Module, these I/O make up the Group 1 signal set. Each signal pair is a 0 to 5VDC input or output. When used as an input or an output a single ended or differential configuration is accommodated.

- Hardware Specifications
  - Environmental Specifications
  - Mechanical Specifications
  - Power Requirements
- Pin Assignments
- Input Specifications
- Input Filtering
- Output Specifications

Hardware Specifications

Environmental Specification

Operating Temperature ..... 0 to 50 degrees C
Storage Temperature ..... -20 to 70 degrees C
Humidity ..... 0 to 90% non-condensing

Mechanical Specification

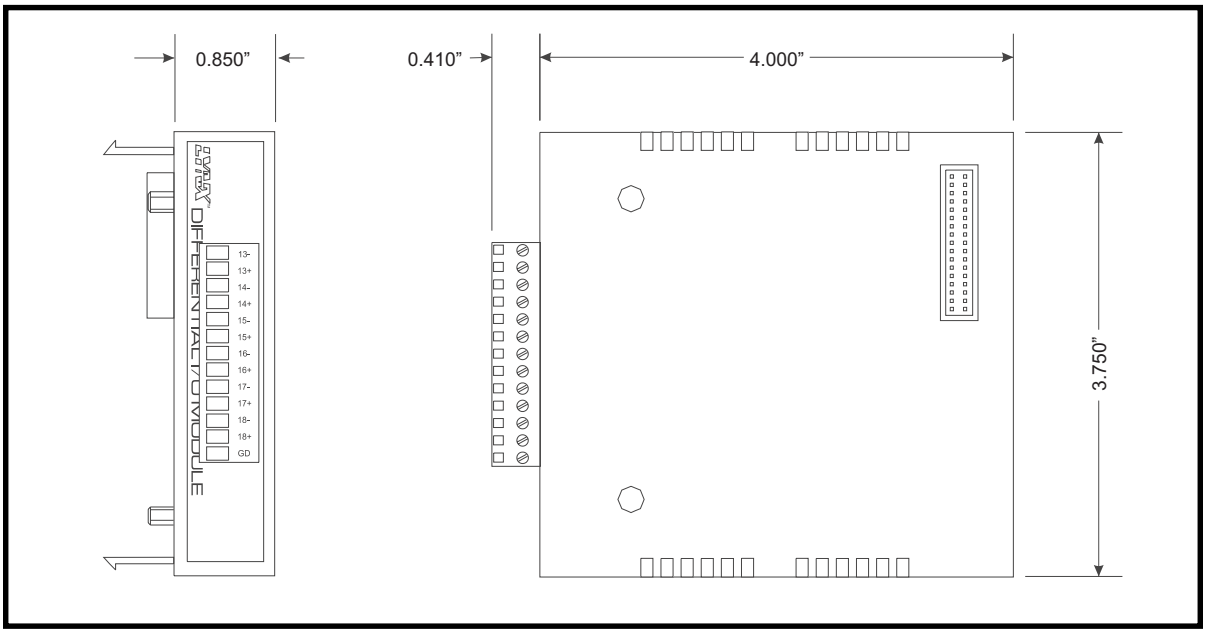


Figure 10.1 : LYNX Differential I/O Module Dimensions

Connection Overview

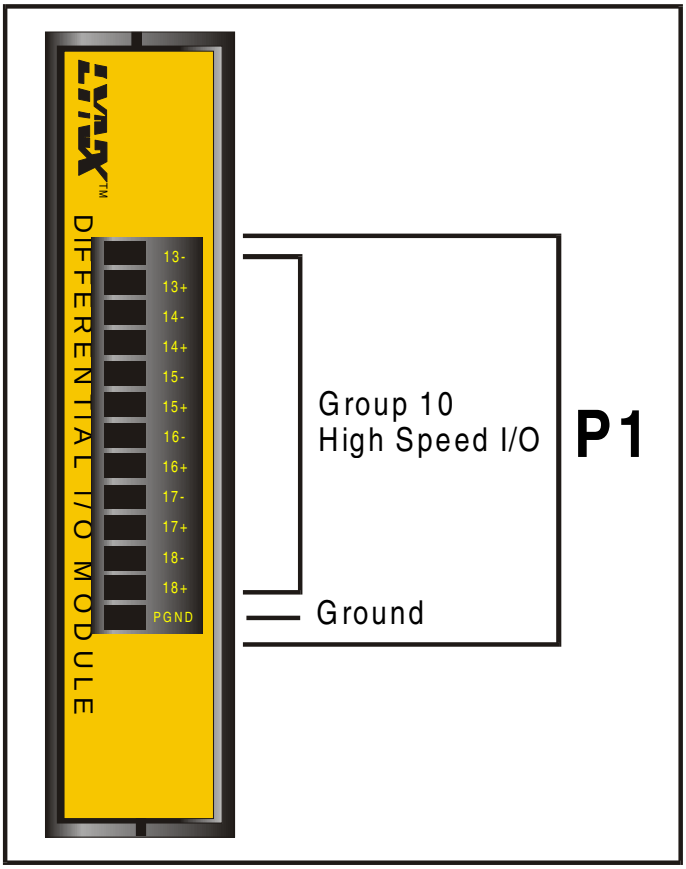


Figure 10.2: High Speed Differential I/O Module Connection Overview

Power Requirements

Power is supplied through the LYNX Control Module.

High Speed Differential I/O Power Requirement	
Input Voltage to LYNX Control Module	**Current Requirement for Module**
+5VDC	50mA
+12 VDC	28mA
+48VDC	8mA
+75VDC	5mA

Table 10.1: High Speed Differential I/O Module Power Requirements

Pin Assingments And Description

P3 - 13 Position Removeable Terminal Connector: High Speed Differential I/O Module		
Pin #	**Function**	**Description**
1	I/O 13-	Pins 1 and 2 are the differentially buffered signal for group 10, I/O 13. This channel is configured by means of the IOS Instruction. This channel is fixed as clock #2 and associated with Counter 2 (CTR2). For usage details see Section 6: Configuring the Digital I/O.
2	I/O 13+	See description above.
3	I/O 14-	Pins 3 and 4 are the differentially buffered signal for group 10, I/O 14.This channel is configured by means of the IOS Instruction. This channel is fixed as clock #2 and associated with Counter 2 (CTR2). For usage details see Section 6: Configuring the Digital I/O.
4	I/O 14+	See description above.
5	I/O 15-	Pins 5 and 6 are the differentially buffered signal for group 10, I/O 15 .This channel is configured by means of the IOS Instruction. This channel is fixed as clock #3 and associated with Counter 3 (CTR3). For usage details see Section 6: Configuring the Digital I/O.
6	I/O 15+	See description above.
7	I/O 16-	Pins 7 and 8 are the differentially buffered signal for group 10, I/O 16. This channel is configured by means of the IOS Instruction. This channel is fixed as clock #3 and associated with Counter 3 (CTR3). For usage details see Section 6: Configuring the Digital I/O.
8	I/O 16+	See description above.
9	I/O 17-	Pins 9 and 10 are the differentially buffered signal for group 10, I/O 17. This channel is configured by means of the IOS Instruction. This Channel may be configured as a high speed input or output. As an output it is a 1MHz reference clock. I/O 17 and 18 are not associated to a counter. For usage details see Section 6: Configuring the Digital I/O.
10	I/O 17+	See description above.
11	I/O 18-	Pins 11 and 12 are the differentially buffered signal for group 10, I/O 18. This channel is configured by means of the IOS Instruction. This Channel may be configured as a high speed input or output. As an output it is a 10MHz reference clock. I/O 17 and 18 are not associated to a counter. For usage details see Section 6: Configuring the Digital I/O.
12	I/O 18+	See description above.
13	PGND	Non-isolated ground. Common with the LYNX Control Module power ground.

Table 10.2: High Speed Differential I/O Module Pin Configuration

Input Specifications

High Speed Differential I/O Input Specifications	
Differential Input Threshold	-0.2V to +0.2V
Input Hysteresis	60mV Typical
Input Common Mode Range	-6V to +6V
Maximum Group Sink	1.5A (Thermally Limited)
Open Circuit Input Voltage + Input	4.3V
Open Circuit Input Voltage - Input	1.4V

Table 10.3: High Speed Differential I/O Module Input Specifications

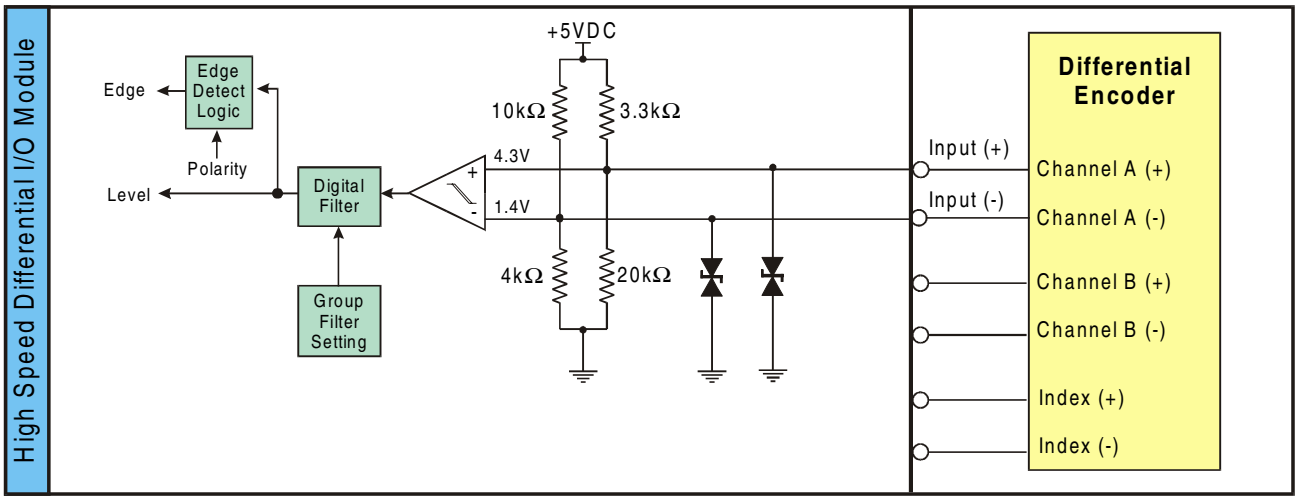


Figure 10.3: LYNX Differential I/O Input Equivalent Circuit

Input Filtering

User definable Digital filtering makes the LYNX well suited for noisy industrial environments. The filter setting is software selectable using the *IOF Variable* with a minimum guaranteed detectable pulse width of 18 microseconds to 2.3 milliseconds.
The table below illustrates the IOF settings.

IOF Filter Settings for the High Speed Differential I/O IOF=<num> (<num> = 0-7)		
Filter Setting	**Cutoff Frequency**	**Minimum Detectable Pulse Width**
0 (default)	5.00 MHz	100 nanoseconds
1	2.50 MHz	200 nanoseconds
2	1.25 MHz	400 nanoseconds
3	625 kHz	800 nanoseconds
4	313 kHz	1.6 microseconds
5	156 kHz	3.2 microseconds
6	78.1 kHz	6.4 microseconds
7	39.1 kHz	12.8 microseconds

Table 10.4: Digital Filter Settings for the Differential I/O

Output Specifications

High Speed Differential I/O Output Specifications		
	No Load	6mA Load
Output Voltage - Logic 0	0.5V	0.8V
Output Voltage - Logic 1	4.5V	4.2V
Short Circuit Current	250mA Max.	

Table 10.5: LYNX Differential I/O Output Specifications

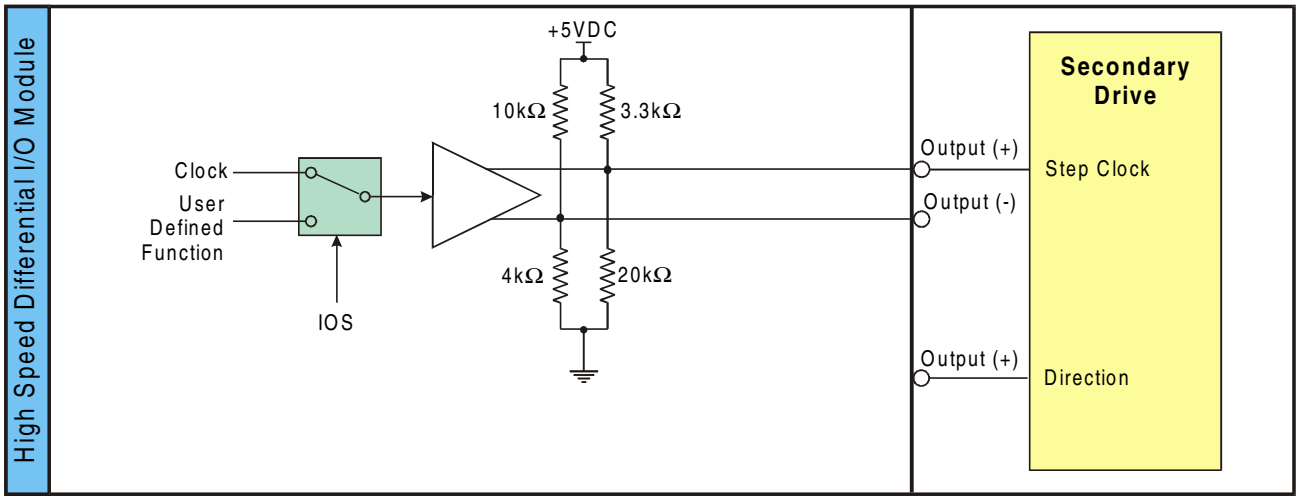


Figure 10.4: LYNX Differential I/O Output Equivalent Circuit

Intentionally Left Blank

MicroLYNX

Part II
MicroLYNX

The MicroLYNX System
Getting Started
Installing and Mounting the MicroLYNX
Powering the MicroLYNX System
Motor Requirements
Controlling the Output Current
The Communications Interface
Configuring and Isolated Digital I/O
Configuring the Using the Expansion Modules

Table of Contents

The MicroLYNX System ***2-7***
Section Overview 2-7
Introduction 2-7
Electrical Specifications 2-8
Power Supply Requirements 2-8
Motor Drive 2-8
Isolated Digital I/O 2-8
Communication Specifications 2-8
Asynchronous 2-8
Controller Area Network (CAN) 2-9
Mechanical Specifications 2-9
Environmental Specifications 2-9
Motion Specifications 2-10
Software Specifications 2-10
Connection Overview 2-11
Getting Started ***2-13***
Section Overview 2-13
Getting Started 2-13
Included in the Package 2-13
User Provided Tools and Equipment Needed 2-14
Connecting the Power Supply 2-14
Motor Connections 2-14
Communications Wiring 2-14
Establishing Communications using the IMS LYNX Terminal 2-14
Testing the MicroLYNX Setup 2-15
Installing and Mounting the MicroLYNX ***2-17***
Section Overview 2-17
Dimensional Information 2-17
Installation and Removal of the Optional Expansion Modules 2-17
Mounting the MicroLYNX System to a Panel 2-19
Powering the MicroLYNX System ***2-20***
Section Overview 2-20
Selecting a Power Supply 2-20
Selecting a Motor Supply (+V) 2-20
Wiring and Shielding 2-21
Rules of Wiring 2-21
Rules of Shielding 2-21
Power Supply Connection & Specification 2-22
Recommended IMS Power Supplies 2-22
Motor Requirements ***2-23***
Section Overview 2-23
Selecting a Motor 2-23
Types and Construction of Stepping Motors 2-23
Sizing a Motor for Your System 2-23
Recommended IMS Motors 2-25
Motor Wiring 2-26
Connecting the Motor 2-27
8 Lead Motors 2-27
6 Lead Motors 2-28
Half Coil Configuration 2-28
4 Lead Motors 2-28
Full Coil Configuration 2-28
Controlling the Output Current and Resolution ***2-29***
Section Overview 2-29

Current Control Variables 2-29
Determining the Output Current 2-30
Setting the Output Current 2-32
Setting the Motor Resolution 2-33
The Communications Interface 2-34
Section Overview 2-34
Connecting the RS-232 Interface 2-34
Connecting the RS-485 Interface 2-38
Connecting and Configuring the Optional Controller Area Network (CAN) Bus 2-40
Connecting to the CAN Bus 2-40
Configuring The CAN Module 2-42
CAN Configuration Command Summary 2-42
To Initialize the CAN Module. 2-43
To Set the CAN Bit Timing Registers 2-44
To Set The Global Mask Registers 2-45
To Setup Message Frames 2-46
Set Message Frame Arbitration Registers 2-46
Defining the MicroLYNX Mode (Single or Party) 2-47
Setting the MicroLYNX Party Address 2-48
MicroLYNX Prompt 2-48
MicroLYNX Baud Rate 2-48
MicroLYNX Modes of Operation 2-49
Immediate Mode 2-49
Program Mode 2-49
EXEC Mode 2-49
MicroLYNX Communication Modes 2-49
ASCII 2-49
Binary 2-50
Configuring the Isolated Digital I/O 2-51
Section Overview 2-51
Electrical Characteristics 2-51
The Isolated Digital I/O 2-51
Uses of the Isolated Digital I/O 2-52
The IOS Variable 2-52
Configuring an Input 2-53
Configuring the Digital Filtering 2-54
Configuring an Output 2-54
The IO Variable 2-55
Read/Write an I/O Group 2-56
Configuring and Using the Expansion Modules 2-57
Section Overview 2-57
MicroLYNX Expansion Modules 2-57
Additional Isolated Digital I/O 2-57
High-Speed Differential I/O Module 2-57
Analog Input/Joystick Module 2-57
Choosing the Expansion Modules for Your Application 2-57
Expanding the Isolated Digital I/O 2-59
Installing The Isolated Digital I/O Module 2-59
Using the Isolated Digital I/O 2-60
The High-Speed Differential I/O Module 2-61
Installing the High-Speed Differential I/O Module 2-62
The Four Clocks Explained 2-62
Clock Types Defined 2-62
Configuring the Differential I/O - The IOS Variable 2-63
Configuring the High Speed I/O a Non-Clock Function 2-64
Configuring an Input 2-64
Setting the Digital Input Filtering for the Differential I/O 2-65

Configuring an Output 2-65
Typical Functions of the Differential I/O 2-66
Connecting and Using an Encoder 2-66
Following an External Clock (Electronic Gearing) 2-68
Configuring the I/O for Half Axis Mode 2-69
One and a Half Axis Operation (RATIOE) 2-70
The Analog Input/Joystick Module 2-71
Installing the Analog Input/Joystick Module 2-71
Instructions & Variables Specific to the Analog Module 2-72
The ADS Variable (A to D Setup) 2-73
Typical Functions of the Analog Input Module 2-74
The RS-232 Port 2 Communication Expansion Module 2-76
The RS-485 Port 2 Communication Expansion Module 2-77

List of Tables

Table 1.1: MicroLYNX Switches 2-11
Table 6.1: Motor Current Control Variables 2-29
Table 6.2: Microstep Resolution Settings 2-33
Table 7.1: Wiring Connections, RS-232 Interface, Single MicroLYNX System 2-35
Table 7.2: Party Mode Address Configuration Switch Settings 2-36
Table 7.3: Connections and Settings, Multiple MicroLYNX System, RS-232 Interface 2-37
Table 7.4: RS-485 Interface Connections 2-38
Table 7.5: Party Mode Address Configuration Switch Settings 2-39
Table 7.6: RS-485 Interface Connections and Settings, Multiple MicroLYNX System 2-37
Table 7.7: CAN Pin Configuration 2-41
Table 7.8: CAN Configuration Command Summary 2-42
Table 7.9: CAN Bit Timing Registers 2-44
Table 7.10: Sample Bit timing Register 2-44
Table 7.11: CAN Bit Time Definition 2-44
Table 7.12: Global Mask Registers 2-44
Table 7.13: Message Frame Arbitration Registers 2-47
Table 7.14: ASCII Mode Special Command Characters 2-50
Table 7.15: Binary Mode Hex Codes 2-50
Table 8.1: IOS Variable Settings 2-53
Table 8.2: Digital Filter Settings for the Isolated I/O 2-54
Table 8.3: Binary State of Outputs 2-56
Table 9.1: Expansion Module Configurations 2-58
Table 9.2: Isolated Digital I/O Group and Line Locations 2-59
Table 9.3: Differential I/O Electrical Characteristics 2-61
Table 9.4: Differential I/O Expansion Module Pinout by Connector Style and Slot 2-61
Table 9.5: The Four Clocks 2-63
Table 9.6: IOS Variable Settings for the Differential I/O 2-64
Table 9.7: Digital Input Filter Settings for the Differential I/O 2-65
Table 9.8: Expansion Slot 2 Encoder Connections 2-66
Table 9.9: Analog Input Module Specifications 2-71
Table 9.10: Analog Input/Joystick Module Pin Configuration 2-71
Table 9.11: Analog Input/Joystick Module Command Summary 2-72
Table 9.12: RS-232 Expansion Module Pinout 2-76
Table 9.13: RS-485 Expansion Module Pinout 2-77

List of Figures

Figure 1.1: Dimensional Information 2-9
Figure 1.2: MicroLYNX Connection Overview 2-11
Figure 1.3 MicroLYNX Switches 2-12
Figure 2.1: Basic Setup Configuration 2-13
Figure 3.1: Dimensional Information 2-17
Figure 3.2: MicroLYNX System with Isolated Digital I/O Expansion Module Installed 2-18
Figure 3.3: Installing the Optional Expansion Modules 2-18
Figure 3.4: Panel Mounting the MicroLYNX 2-19
Figure 4.1: MicroLYNX Power Connections 2-22
Figure 5.1: Per Phase Winding Inductance 2-24
Figure 5.2: 8 Lead Motor, Series Connection 2-27
Figure 5.3: 8 Lead Motor, Parallel Connection 2-27
Figure 5.4: 6 Lead Motor, Half Coil Connection 2-28
Figure 5.5: 6 Lead Motor, Full Coil Connection 2-28
Figure 5.6: 4 Lead Motor 2-28
Figure 6.1: Motor Current Control Variables 2-29
Figure 7.1: Connecting the RS-232 Interface, Single MicroLYNX System 2-35
Figure 7.2: Connecting the RS-232 Interface, Multiple MicroLYNX System 2-37
Figure 7.3: RS-485 Interface, Single MicroLYNX System 2-38
Figure 7.4: RS-485 Interface, Multiple MicroLYNX System 2-40
Figure 7.5: Devices on a CAN Bus 2-41
Figure 7.6: Connecting the CAN Bus 2-41
Figure 7.7: Bit Register Configuration Dialog from LYNX Terminal 2-45
Figure 7.8: Setup Dialog for Global Mask Registers in LYNX Terminal 2-46
Figure 7.9: Message Frame Setup Dialog From LYNX Terminal 2-47
Figure 7.10: LYNX CAN Setup Dialog from LYNX Terminal 2-48
Figure 8.1: IOS Variable Applications 2-52
Figure 8.2: Isolated Digital I/O Input Equivalent Circuit 2-53
Figure 8.3: Isolated Digital I/O Output Equivalent Circuit 2-55
Figure 9.1: Installing the Isolated I/O Module 2-59
Figure 9.2: The Isolated I/O Expansion Module, Bottom View 2-60
Figure 9.3: Powering Multiple Isolated Digital I/O Modules 2-60
Figure 9.4: The Differential I/O Module 2-61
Figure 9.5: Installing the High-Speed Differential I/O Module 2-62
Figure 9.6: Clock Functions 2-63
Figure 9.7: Differential I/O Input Equivalent Circuit 2-64
Figure 9.8: Differential I/O Output Equivalent Circuit 2-65
Figure 9.9: Differential Encoder Connection 2-67
Figure 9.10: Differential I/O Connections for Following and External Clock Input 2-69
Figure 9.11: One and a Half Axis Operation 2-70
Figure 9.12: Installing the Analog Input/Joystick Module 2-72
Figure 9.13: Analog Input Example Connection 2-74
Figure 9.14: Connecting the RS-232 Expansion Module 2-76
Figure 9.15: Connecting the RS-485 Expansion Module 2-77

SECTION 1

The MicroLYNX System

Section Overview

This section summarizes the specifications for the basic MicroLYNX system. It contains the following:

- Introduction
- Electrical Specifications
- Mechanical Specifications

Introduction

The MicroLYNX is a Motion Control System integrating a bipolar stepper motor microstepping drive and a programmable indexer with expandable I/O and communication capability into a compact panel mounted assembly. The basic system is available with either 3 Amp (MicroLYNX-4) or 5 Amp (MicroLYNX-7) RMS motor drive capability. The basic MicroLYNX System can also be purchased with either the standard dual communications or Controller Area Network (CAN) interface.

The MicroLYNX has been developed from the "LYNX 1.5 Axis Modular Motion System". It has inherited all of its capabilities along with enhanced features and additional software commands to make use of these features and control the motor drive parameters.

The integration of the drive and the small size of the MicroLYNX are the most obvious accomplishments in its development. The ability to customize the I/O suite to the application in smaller increments is another. The basic MicroLYNX System comes standard with six +5 to +24VDC isolated digital programmable I/O lines. This is epandable to a total of twenty-four lines using optional expansion modules. This section summarizes the specifications for the basic MicroLYNX system. The expansion modules available for the MicroLYNX are:

- Isolated Digital +5 to +24VDC I/O
- High Speed Differential I/O
- Analog Input / Joystick Interface
- RS-232 Module (for use with the CAN version only)
- RS-485 Module (for use with the CAN version only)

These modules and their applications are covered in detail in *Section 9: Configuring and Using the Expansion Modules.*

A more subtle enhancement is the provision of two fully independent communication ports for the LYNX system. While modular LYNX provided both RS-232 and RS-485 ports, these ports shared the same UART on the LYNX CPU. This limited communications on these ports to sequential usage.

Adding a fully independent second UART allows simultaneous usage. Software has been updated to keep this system fully compatible with the Modular LYNX. The MicroLYNX will accept commands from either COMM port and can now direct output to either port regardless of the state of the HOST flag. Of course, compatibility means that HOST mode is still supported. A motion system architecture might use one COMM port for connection to a host PC or PLC while using the other for communication with an operator interface or status display. Another use for the second port could be to pass data between MicroLYNX Systems in a multi-axis system while maintaining a communications link to a host.

Electrical Specifications

Power Supply Requirements

Voltage

-4 Version (P/N MX-CS100-400) ..... +12 to +48VDC
-7 Version (P/N MX-CS100-700) ..... +24 to +75VDC

Current

Actual requirements depend on application and programmable current setting.

-4 Version (P/N MX-CS100-400) ..... 2A typical, 4A peak
-7 Version (P/N MX-CS100-700) ..... 3A typical, 6A peak

Motor Drive

Motor type ..... 2/4 phase bipolar stepper
Motor Current (software programmable)
-4 Version (P/N MX-CS100-400) ..... to 4A peak
-7 Version (P/N MX-CS100-700) ..... to 7A peak
MicroStep Resolution (# of settings) 14
Steps per Revolution (1.8° Motor)
400, 800, 1000, 1600, 2000, 3200, 5000, 6400, 10000, 12800, 25000, 25600, 50000, 51200

General Purpose I/O

Number of I/O 6
Input Voltage ..... +5 to +24VDC
Output Current Sink 350mA
Input Filter Range ..... 215Hz to 21.5kHz (Programmable)
Pull-ups ..... 7.5kOhm individually switchable
Pull-up Voltage
Internal ..... +5VDC
External ..... +24 VDC
Protection ..... Over temp, short circuit, inductive clamp
Isolated Ground ..... Common to the 6 I/O

Communication Specifications

Asynchronous

Interface Type:
COMM 1 ..... RS-232
COMM 2 ..... RS-485
# of Bits / Character 8
Parity ..... None
Handshake ..... None
BAUD Rate ..... 4800 to 38.8kbps
Error Checking ..... 16 bit CRC (binary mode)
Communication Modes ..... ASCII text or binary
Isolated Ground ..... Common to COMM 1 and COMM 2

Controller Area Network (CAN)

CAN replaces Asynchronous Communications in the MicroLYNX Base System. (Uses COMM 1 internally.)

CAN Compliance ..... Version 2.0B Active
Message Frames
Receive 2
Transmit 1
Isolated Ground ..... Common to COMM 1 and COMM 2

Mechanical Specifications

Dimensions ..... See Figure 1.1
# of Expansion Modules 3
Cooling ..... Built in fan
Mounting ..... 2 #6 (or M3.5) machine screws
Mounting Screw Torque ..... 5 to 7 lb-in

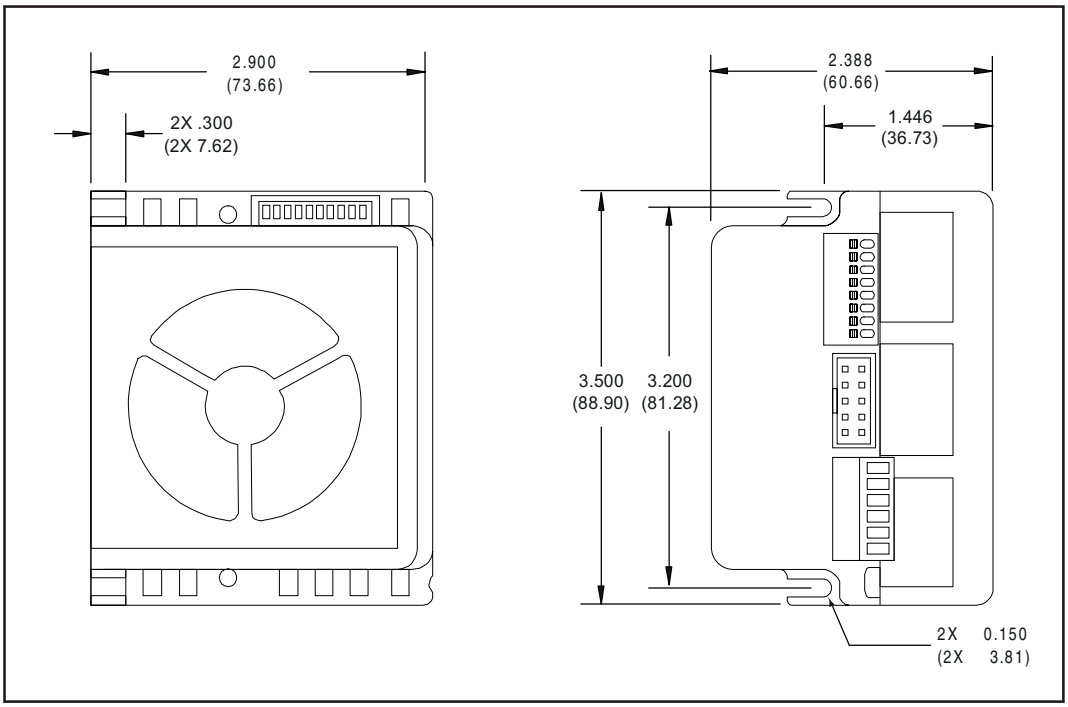


Figure 3.1: Dimensional Information

Environmental Specifications

Operating Temperature ..... 0 to 50°C
Storage Temperature ..... -20 to 70°C
Humidity ..... 0 to 90% non-condensing

Motion Specifications

Counters

Type ..... Position, encoder #1, encoder #2
Resolution ..... 32 bits
Edge Rate (Max) ..... 5 MHz

Electronic Gearing

Use of Electronic Gearing requires the Differential I/O Expansion Module.

Range* ..... -1 to 1 (external clock in)
Resolution ..... 32 bits
Range* ..... -2 to 2 (secondary clock out)
Resolution ..... 16 bits
*The range may be increased by adjusting the microstep resolution of the drive.

Velocity

Range ..... ±5,000,000 steps/sec
Resolution ..... 0.005 steps/sec
Update Period ..... 25.6 microseconds

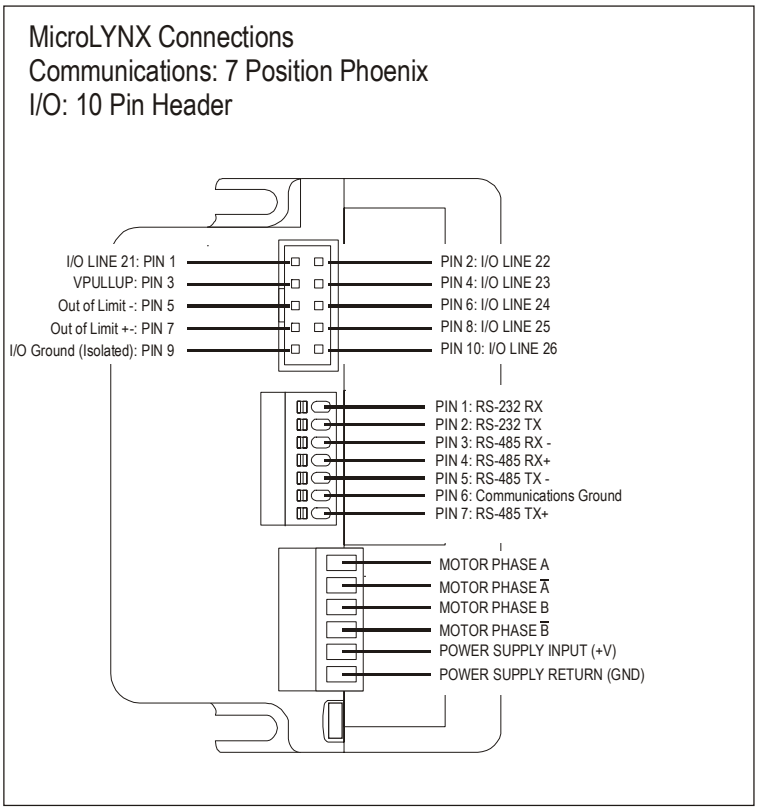
Acceleration/Deceleration

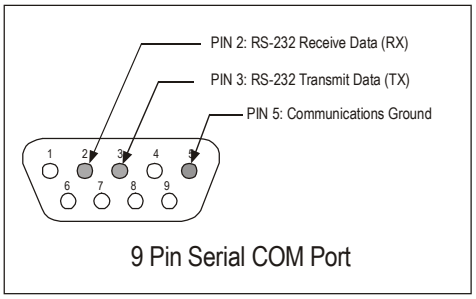
Range ..... ±1,530,000,000 steps/sec^2
Resolution ..... 0.711 steps/sec^2
Types:
Linear, triangle s-curve, parabolic, sinusoidal s-curve, user-defined.

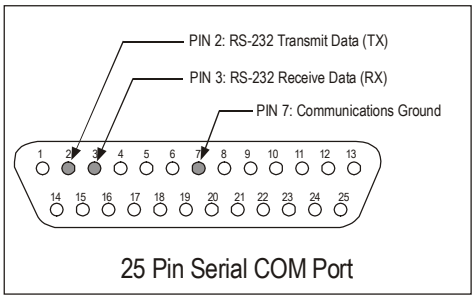
Software Specifications

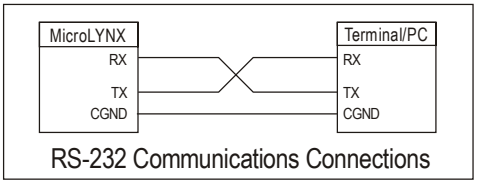
User Program Space ..... 8175 bytes
Number of User Definable Labels 291
Program and Data Storage ..... Flash
Math, Logic, and Conditional Functions
(32 Bit Floating Point Math IEEE Format):
Add, Subtract, Multiply, Divide, Sine, Cosine, Tangent, Arc Sine, Arc Cosine, Arc Tangent, AND, OR, XOR, NOT, Less Than, Greater Than, Equal, Square Root, Absolute, Integer Part, Fractional Part
Acceleration & Deceleration:
Separate Variables and Flags. 4 Pre-defined Types and 1 User-defined
Limit Switch ..... Definable: Deceleration & Type
Isolated I/O ..... Programmable as Dedicated or General Purpose
Predefined I/O Functions ..... 25 (Limit, Home, Soft Stop, etc.)
Program Trip Functions 13
(4 I/O Input Trips, 4 Timer Trips, 4 Position Trips, 1 Velocity Trip)
User Programs:
2 Executed simultaneously: 1 Foreground, 1 Background.
Party Mode Names 62
Communication Modes ..... 2: ASCII, Binary
Mechanical Compensation ..... Backlash
Encoder Functions ..... Stall Detection and Position Maintenance

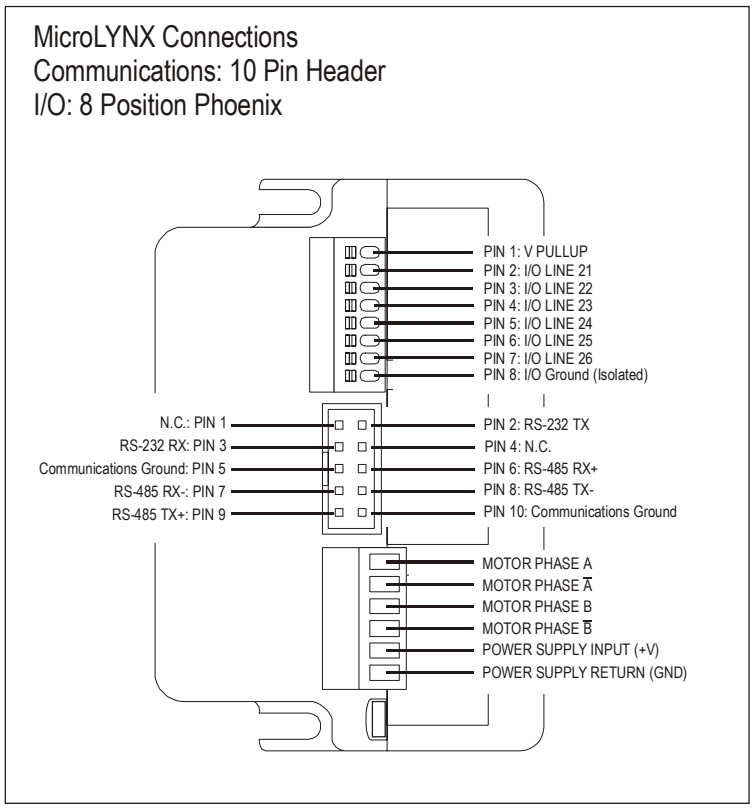
Connection Overview

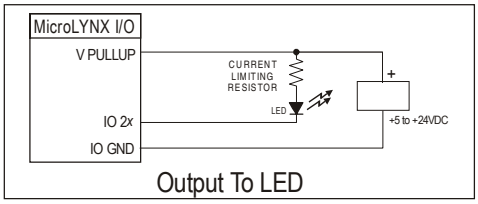


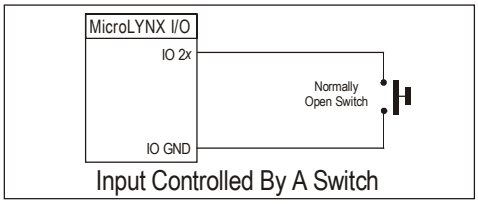












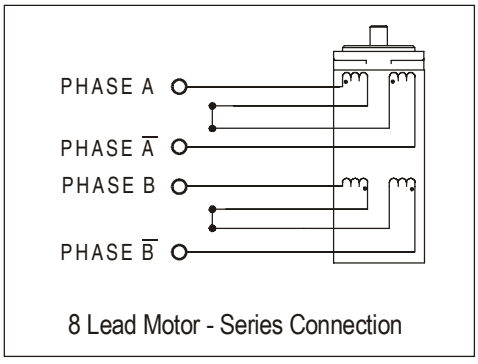


Figure 1.2: MicroLYNX Connection Overview

Switches

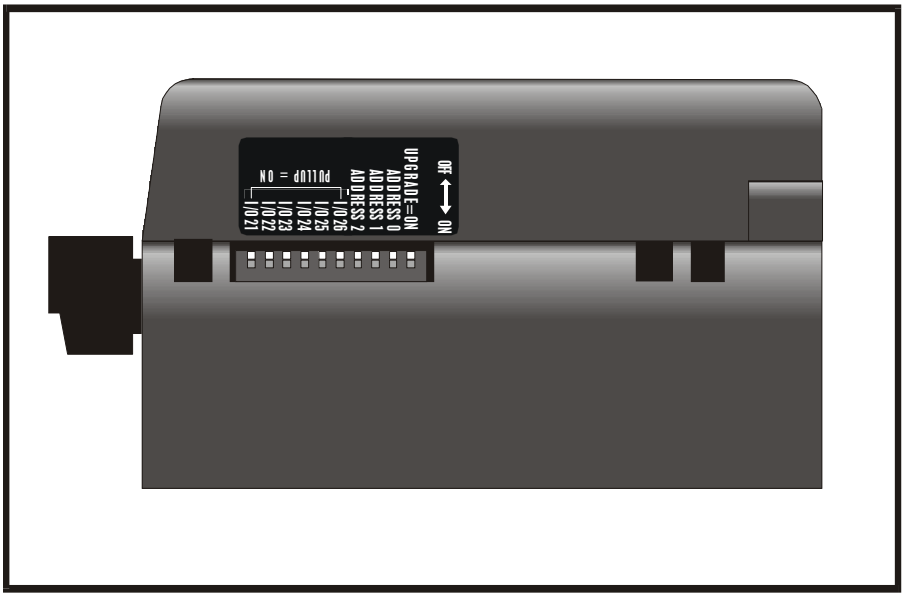


Figure 1.3: MicroLYNX Switches

MICROLYNX SWITCHES		
SWITCH #	**Switch Name**	**Function**
1-6	I/O 26	Pull-up ON/OFF Switches for I/O Lines 21-26
7-9	Address 2-0	Multi-drop Communications Address. (Also can be configured in software.)
10	Upgrade	Firmware Upgrade

Table 1.1: MicroLYNX Switches

SECTION 2

Getting Started

Section Overview

The purpose of this section is to get you up and running quickly. This section will help you do the following:

- Connect power to the MicroLYNX Control System.
- Connect and establish communications in single mode.
- Write a simple test program.

Getting Started

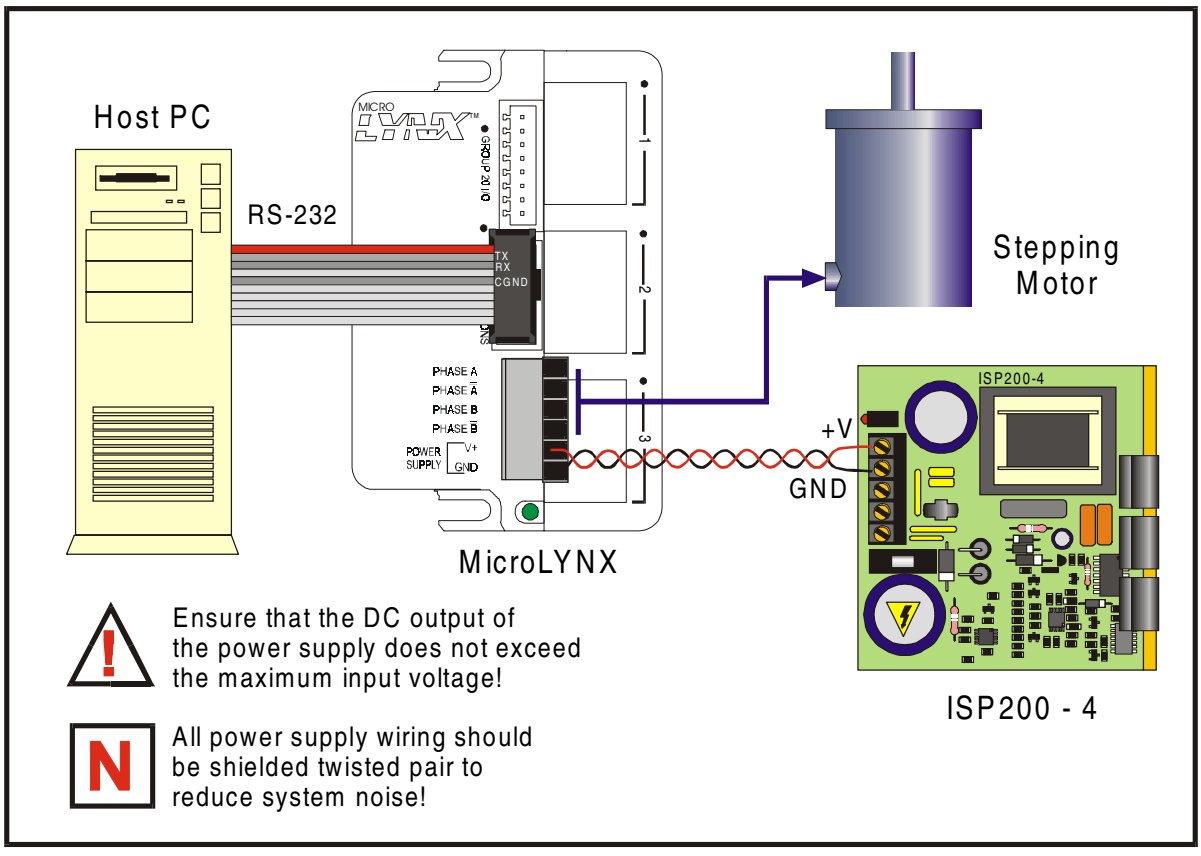


Figure 2.1: Basic Setup Configuration, RS-232 Interface

Included in the Package

(1) MicroLYNX Controller ..... IMS P/N MX-CS100-400
(1) LYNX/MicroLYNX Compact Disc ..... IMS P/N LX-SW100-000
(1) Quick Manual ..... IMS P/N MX-OM300-000

User Provided Tools and Equipment Needed

- Serial Cable.
- ISP200-4 or equivalent power supply.
- M2-22XX or equivalent stepping motor.
- Wire Cutters/Strippers.
- 22 gauge wire for logic level signals.
- 18 gauge wire for power supply and motor wiring.
- PC with a free serial port (COM 1 or 2).

Connecting the Power Supply

1. Using the 18 gauge wire, connect the DC output of your power supply to V+ on your MicroLYNX Control System. See *Figure 2.1: Basic Setup Configuration* for detials.
2. Connect the Power Supply Return (GND) to GND on the MicroLYNX Controller.
3. Connect the AC Line cord to your power supply in accordance with any user documentation. **DO NOT PLUG IN AT THIS TIME!**

Motor Connections

Connect the motor to the MicroLYNX System in accordance with *Figure 2.1*.

Communications Wiring

Connect the Host PC to the MicroLYNX in accordance with *Figure 2.1*. This is needed to program the MicroLYNX. If the MicroLYNX has a Terminal Block Connector, connect in the following manner:

PC(25 Pin Serial Port)	PC(9-Pin Serial Port)	MicroLYNX Comm Connector
Pin 7 (GND)	Pin 5 (GND)	Pin 6 (C GND)
Pin 2 (TX)	Pin 3 (TX)	Pin 1 (RS-232 RX)
Pin 3 (RX)	Pin 2 (RX)	Pin 2 (RS-232 TX)

Establishing Communications using the IMS LYNX Terminal

Included in the MicroLYNX shipping package is the IMS LYNX Terminal software. This is a programming/ communications interface created by IMS to simplify the use of the MicroLYNX. There is a 16 bit version for Windows 3.x and a 32 bit version for Windows 9x/NT4/2000 located on the CD. The IMS LYNX Terminal is also necessary to upgrade the software in your MicroLYNX. These updates will be posted to the IMS website at http://www.imshome.com/ as they are made available.

To install the IMS LYNX Terminal to your hard drive, insert the CD into your CD-ROM Drive. The 3.5" CD, while smaller than typical compact disks, will work in any tray-type CD drive. For Windows 3.x users click "File > Run" in program manager and type "x:\terminal\16bit\setup.exe" in the "Command Line" box (x:\ is the drive letter of your CD-ROM). For Windows 9x/NT4/2000 users click "Start > Run" and type "x:\terminal\32bit\setup.exe" in the "Open" box.

Follow the on-screen instructions to complete the installation.

1) Open the LYNX Terminal by double clicking the IMS icon labeled "LYNXTERM" in the "LynxTerm" program group (Windows 3.x) or select Start>Programs>LynxTerm>LYNXTERM (Windows 9x/NT/2000).
2) Click the File Menu Item "Setup".
3) Select the "Terminal>Setup" option.
4) Select the Communications Port that you will be using with your MicroLYNX.

5) The BAUD rate is already set to the MicroLYNX default. Do not change this setting until you have established communications with the MicroLYNX Controller.

6) The "Window Size" settings are strictly optional. You may set these to whatever size is comfortable to you.

7) Click "OK". The settings will be automatically saved upon a normal shutdown.

8) Apply power to the MicroLYNX Controller. The following sign-on message should appear in the Terminal window:

```
Program Copyright © 1996-2000 by:
Intelligent Motion Systems, Inc.
Marlborough, CT 06447
VER =       SER =
```

Detailed instructions for the IMS LYNX Terminal software can be located in The LYNX / MicroLYNX Software Reference Manual.

Testing the MicroLYNX Setup

Two basic instructions for communicating with a LYNX Product are SET and PRINT. The SET instruction is assumed and can be left off when communicating in ASCII mode. (You are in ASCII mode whenever you are using a text based terminal.) It is used to set variables and flags that define LYNX operation. The MicroLYNX Software automatically recognizes the SET instruction whenever the name of the variable or flag is typed into the terminal. Here we will set the motor units variable (MUNIT) to 51200 by typing the following at the prompt (>):

```
MUNIT = 51200
```

The PRINT instruction is used to report the values of variables and flags. Now double-check the value of MUNIT by typing the following at the prompt (>):

```
PRINT MUNIT
```

The return from your terminal should be 51200. Note that the case is not important for instructions, variables and flags. They may be typed in upper or lower case.

The next step is to turn the motor. Before doing so, type in the following lines to configure the driver at the prompt. (Note: Command descriptions are found in the Software Reference section of the manual.)

MSEL = 256
MAC = 80
MRC = 75

Use the SLEW instruction to move the motor at a constant velocity. Be sure that the velocity provided is a reasonable value for your motor and drive and try to move the motor. For instance, at the prompt type:

```
SLEW 10
```

This will move the motor at a speed of 10 munits per second. If the motor does not move, verify that the wiring is in accordance with *Figure 2.1*. If the wiring is determined to be correct, type:

```
PRINT ERROR
```

An error number other than zero (0) will be displayed. See *Appendix B* for more information.

Once you have been able to move the motor, the next step is to write a simple program to illustrate one of the dynamic features of the MicroLYNX: the ability to convert motor steps to a dimension of linear or rotary distance. Let's begin by discussing the relationship between the MUNIT variable and user units. Typically, when we perform a move, we want to know the distance of that move in a familiar unit of measurement. That means translating motor steps to the desired unit of measurement. The MicroLYNX Controller has the capability of doing this for you. You have already set the motor units variable (MUNIT) to a value of 51200.

With the driver set to a resolution of 256 microsteps per step (MicroLYNX factory default) and a 1.8° step motor that will be equal to 1 revolution of the motor, or one USER UNIT. A user unit can be any unit of measure. At this point, by entering the instruction MOVR 1, the motor will turn one complete revolution relative to its current position. Therefore, 1 User Unit = 1 Motor Revolution. For the exercise below we will use degrees for our user unit. As the MicroLYNX Product Manual indicates, the calculation required to select degrees as our user unit in this case is:

51200 Microsteps per rev ÷ 360 degrees = 142.222 Microsteps per degree

By setting the MUNIT variable to 51200/360, the MicroLYNX will perform the calculation to convert the user unit to degrees.

Now, when a relative motion instruction "MOVR 90" is issued the motor will turn 90 degrees.

Let's enter a sample program that will convert motor steps to degrees, execute a 90° move and report that move every 100 milliseconds while the motor is moving. Type the following bold commands:

```
'Enter Program Mode, start program at Location 2000.
PGM 2000
'Label the program TSTPGM.
LBL TSTPGM
'Set the user units to degrees.
MUNIT = 51200/360
'Set the max. velocity to 25 degrees per second.
VM = 25
'Execute a relative move of 90 degrees.
MOVR 90
'Report the position every 100 ms while moving.
LBL PRINTPOS
DELAY 100
PRINT "Axis position is", POS, "Degrees."
BR PRINTPOS, MVG
'End the program.
END
PGM
```

Now Type **TSTPGM** to run program.
This sample program will be stored starting at location 2000. It sets the conversion factor for the user units, sets the maximum velocity and then starts a motion. While the motion is occurring, the position is reported every 100 milliseconds.

At this point you may desire to restore the settings to their factory default as you may not wish to use degrees as your user unit. To do this, you will use the CP, DVF and IP instructions.

CP - Clear Program.
To clear the program, type CP 1, 1. This will completely clear program memory space. Should you desire to only remove one program, the instruction "CP [Program Label]", i.e. "CP TSTPGM", would clear only the specified program. In this exercise only one program was entered, "CP TSTPGM" will clear it.

DVF - Delete User-Defined Variables and Flags.
By entering DVF, all of the user defined variables will be removed. Although no flags were set in this exercise, this command would clear them were they used.

IP - Initialize Parameters.
This instruction will restore all of the parameters to their factory default state.

After entering these instructions, a SAVE instruction should be entered.

SECTION 3

Installing and Mounting the MicroLYNX

Section Overview

This section covers the installation of the optional expansion modules and panel mount procedures for the MicroLYNX System.

- Dimensional information.
- Installation and removal of the optional expansion modules.
- Mounting the MicroLYNX System to a panel.

Dimensional Information

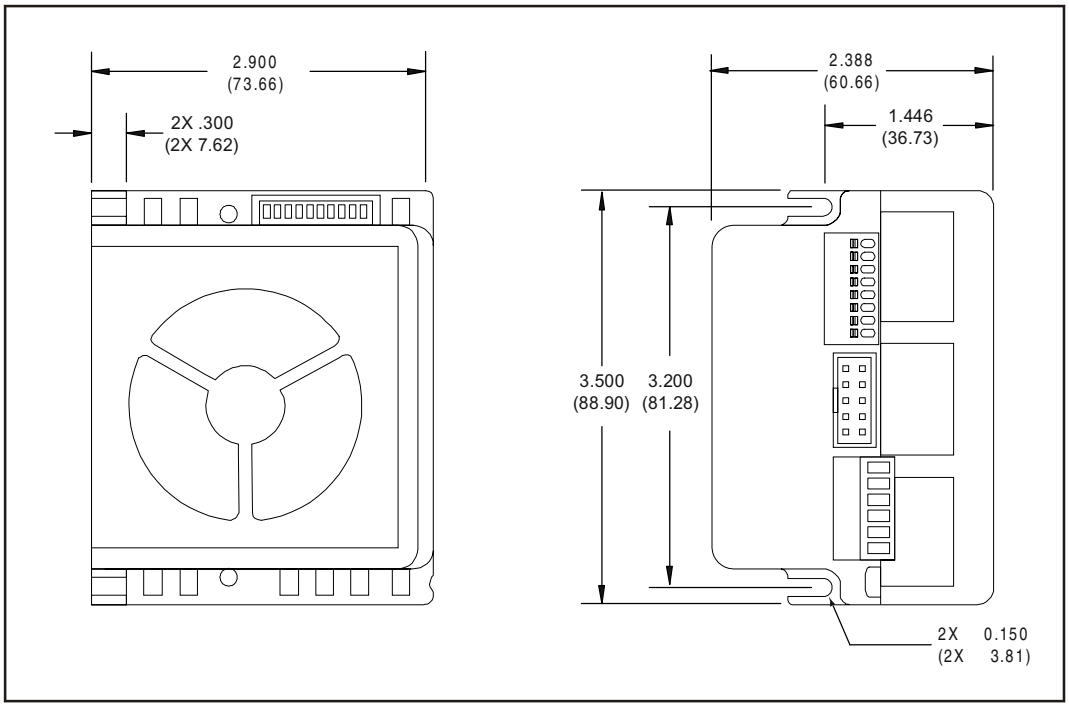


Figure 3.1: Dimensional Information

Installation and Removal of the Optional Expansion Modules

One of the powerful features of the MicroLYNX System is the extreme ease by which it can be configured and installed. There are three (3) bays in which expansion modules can be installed. The expansion modules can be plugged into any available slot, with the exception of the High Speed Differential I/O modules, which can only be plugged into slots 2 and 3. For ease of configuration, ensure that the pull-up switches on the Isolated Digital I/O expansion module are in the desired position prior to closing and mounting the

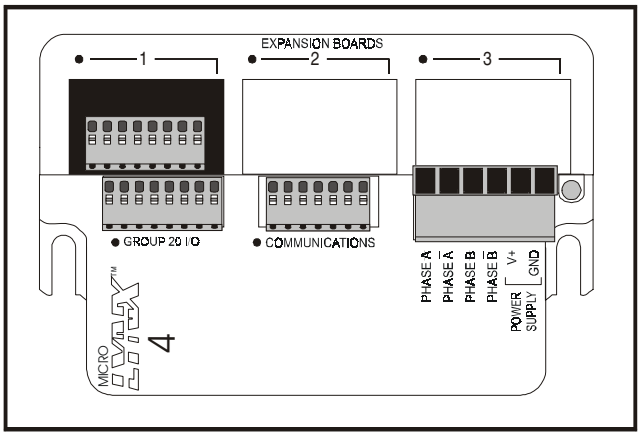


Figure 3.2: MicroLYNX System showing the General Purpose I/O Expansion Module installed in Slot #1

MicroLYNX System. See *Section 9: Configuring and Using the Optional Expansion Modules* for more information on this topic.

To install the expansion modules, the only tool required is a phillips head screwdriver. The installation steps follow:

1) Remove the two screws [A] from the MicroLYNX case.
2) Remove the side of the case. (*See Figure 3.3*).
3) Remove the cover from the slot you will be using.
4) Insert Expansion Module into open slot, seating until module snaps into place. (*See Figure 3.3*)
5) Replace the side of the case.
6) Insert and tighten screws.

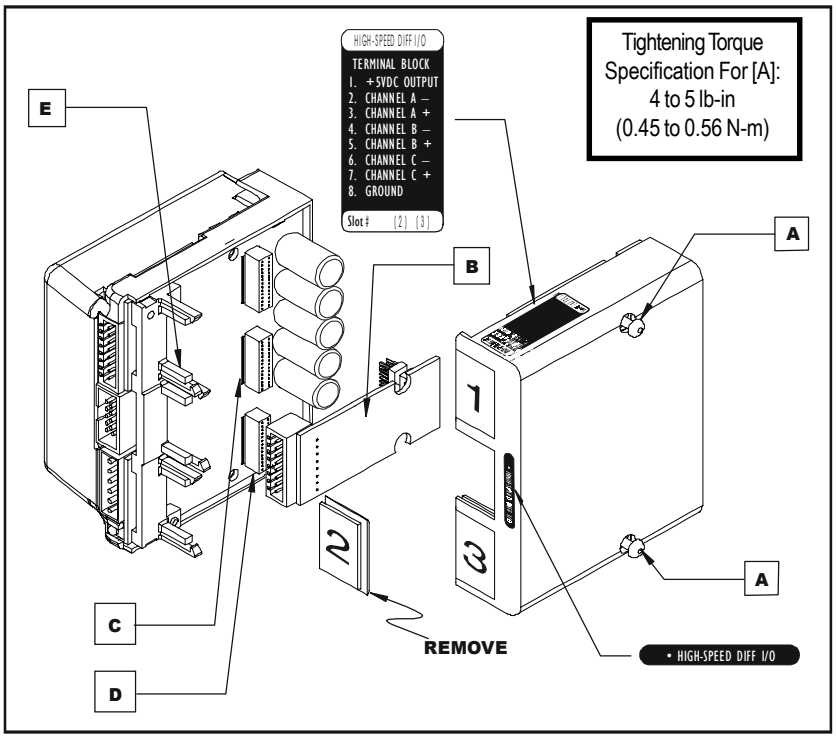


Figure 3.3: Installing the Optional Expansion Modules

Mounting the MicroLYNX System to a Panel

The MicroLYNX System can be mounted to a panel by using standard #6 hardware. As the system has a built-in cooling fan, no heat sinking is necessary. When mounting the MicroLYNX System in an enclosure, ensure that adequate space is available for air flow on the fan side of the MicroLYNX case. Mounting screws should be tightend to 5-7 lb-in torque.

WARNING! Ensure that adequate clearance on the fan side of the case is left for air flow.

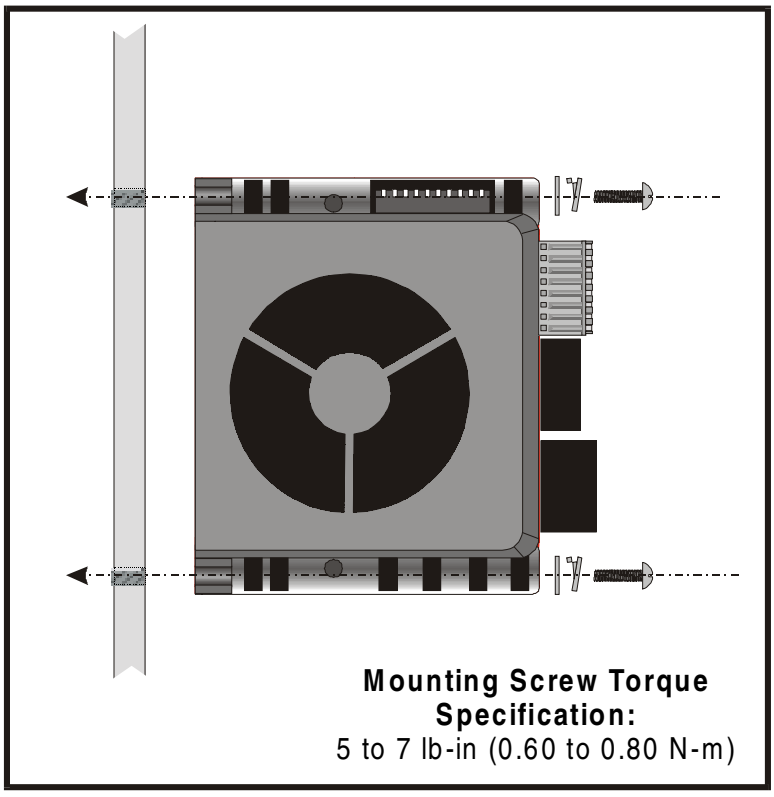


Figure 3.4: Panel Mounting the MicroLYNX

SECTION 4

Powering the MicroLYNX System

Section Overview

This section covers the power requirements for your MicroLYNX System.

- Selecting a power supply.
- Basic rules of wiring and shielding.
- Power supply connection and requirements.
- Recommended power supplies.

Selecting a Power Supply

Selecting a Motor Supply (+V)

Proper selection of a power supply to be used in a motion system is as important as selecting the drive itself. When choosing a power supply for a stepping motor driver, there are several performance issues that must be addressed. An undersized power supply can lead to poor performance, and possibly even damage, to your MicroLYNX System.

The Power Supply - Motor Relationship

Motor windings can be basically viewed as inductors. Winding resistance and inductance result in an L/R time constant that resists the change in current. To effectively manipulate the rate of charge, the voltage applied is increased. When traveling at high speeds there is less time between steps to reach current. The point where the rate of commutation does not allow the driver to reach full current is referred to as Voltage Mode. Ideally you want to be in Current Mode, which is when the drive is achieving the desired current between steps. Simply stated, a higher voltage will decrease the time it takes to charge the coil and, therefore, will allow for higher torque at higher speeds.

Another characteristic of all motors is back EMF. Back EMF is a source of current that can push the output of a power supply beyond the maximum operating voltage of the driver and, as a result, could damage the stepper driver over a period of time.

The Power Supply - Driver Relationship

The MicroLYNX System is very current efficient as far as the power supply is concerned. Once the motor has charged one or both windings of the motor, all the power supply has to do is replace losses in the system. The charged winding acts as an energy storage in that the current will recirculate within the bridge, and in and out of each phase reservoir. This results in a less than expected current draw on the supply. Stepping motor drivers are designed with the intention that a user's power supply output will ramp up to greater or equal to the minimum operating voltage. The initial current surge is quite substantial and could damage the driver if the supply is undersized. The output of the power supply could fall below the operating range of the driver upon a current surge if it is undersized. This could cause the power supply to start oscillating in and out of the voltage range of the driver and result in damage to either the supply, the driver or both. There are two types of supplies commonly used, regulated and unregulated, both of which can be switching or linear. All have their advantages and disadvantages.

Regulated vs. Unregulated

An unregulated linear supply is less expensive and more resilient to current surges; however, the voltage decreases with increasing current draw. This can cause problems if the voltage drops below the working range of the drive. Also of concern are the fluctuations in line voltage. This can cause the unregulated linear supply to be above or below the anticipated or acceptable voltage.

A regulated supply maintains a stable output voltage, which is good for high speed performance. They are also not bothered by line fluctuations; however, they are more expensive. Depending on the current regulation, a regulated supply may crowbar or current clamp and lead to an oscillation that, as previously stated, can cause damage to the driver and/or supply. Back EMF can cause problems for regulated supplies as well. The current regeneration may be too large for the regulated supply to absorb. This could lead to an over voltage condition which could damage the output circuitry of the MicroLYNX System. Non IMS switching power supplies and regulated linear supplies with overcurrent protection are not recommended because of their inability to handle the surge currents inherent in stepping motor systems.

Wiring and Shielding

Noise is always present in a system that involves high power and small signal circuitry. Regardless of the power configuration that you use in your system, there are some wiring and shielding rules that you should follow to keep your noise-to-signal ratio as small as possible.

Rules of Wiring

- Power Supply and Motor wiring should be shielded twisted pair run separately from signal carrying wires.
- A minimum of 1 twist per inch is recommended.
- Motors wiring should be shielded twisted pairs using 20 gauge wire, or 18 gauge or better for distance greater than 5 feet.
- Power ground return should be as short as possible to established ground.
- Power Supply wiring should be shielded twisted pairs. Use 18 gauge wire if load is less than 4 amps, or 16 gauge for more than 4 amps.
- Do not “Daisy-Chain” power wiring to system components.

Rules of Shielding

- The shield must be tied to zero-signal reference potential. In order for shielding to be effective, it is necessary for the signal to be earthed or grounded.
- Do not assume that earth ground is true earth ground. Depending on the distance to the main power cabinet, it may be necessary to sink a ground rod at a critical location.
- The shield must be connected so that shield currents drain to signal-earth connections.
- The number of separate shields required in a system is equal to the number of independent signals being processed plus one for each power entrance.
- The shield should be tied to a single point to prevent ground loops.
- A second shield can be used over the primary shield; however, the second shield is tied to ground at both ends.

WARNING! When using an unregulated supply, ensure that the output voltage does not exceed the maximum driver input voltage due to variations in line voltage! It is recommended that an input line filter be used on power supply to limit voltage spikes to the system!

Power Supply Connection & Specification

Power is connected to the MicroLYNX via connector P1. All optional expansion boards are then powered from the MicroLYNX.

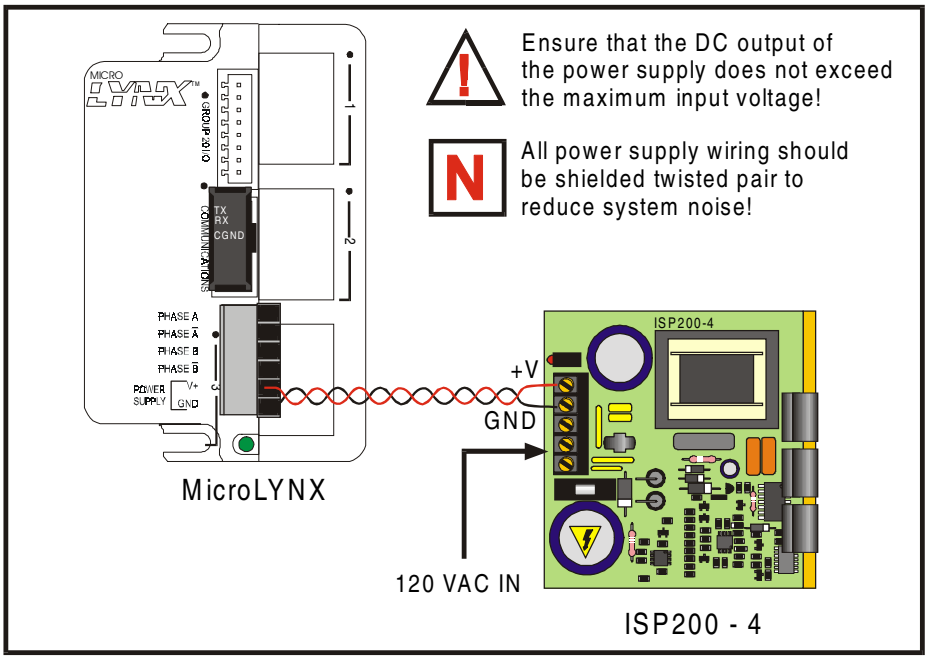


Figure 4.1: MicroLYNX Power Connection

Power Supply Specifications

Recommended Type ..... Unregulated DC
Ripple Voltage ..... ±10%

MicroLYNX - 4

Output Voltage ..... +12 to +48VDC
*Output Current ..... 2 Amps (Typical) 4 Amps (Peak)

MicroLYNX - 7

Output Voltage ..... +24 to +75VDC
*Output Current ..... 3 Amps (Typical) 6 Amps (Peak)

*The output current needed is dependant on the supply voltage, motor selection and load.

Recommended IMS Power Supplies

The ISP200 is a low-cost non-regulated switching power supply which can handle varying load conditions. It is available in either 120 or 240 VAC configuration.

ISP200-4(MicroLYNX-4)/ISP200-7(MicroLYNX-7)

Input Specifications

*AC Input Voltage Range 102-132VAC
Frequency ..... 50-60 Hz

Output Specifications

Voltage (Nominal - No Load) 45 VDC/75VDC
Current (Continuous) ..... 3/2 Amps

***Options**

ISP200H-4/7 ..... 240VAC Input

SECTION 5

Motor Requirements

Section Overview

This section covers the motor configurations for the MicroLYNX-4/7.

- Selecting a motor.
- Motor wiring.
- Connecting the motor.

Selecting a Motor

When selecting a stepper motor for your application, there are several factors that need to be taken into consideration.

- How will the motor be coupled to the load?
- How much torque is required to move the load?
- How fast does the load need to move or accelerate?
- What degree of accuracy is required when positioning the load?

While determining the answers to these and other questions is beyond the scope of this document, they are details that you must know in order to select a motor that is appropriate for your application. These details will effect everything from the power supply voltage to the type and wiring configuration of your stepper motor, as well as the current and microstepping settings of your MicroLYNX System.

Types and Construction of Stepping Motors

The stepping motor, while classed as a DC motor, is actually an AC motor that is operated by trains of pulses. Though it is called a “stepping motor”, it is in reality a Polyphase Synchronous Motor. This means it has multiple phases wound in the stator and the rotor is dragged along in synchronism with the rotating magnetic field. The MicroLYNX System is designed to work with the following types of stepping motors:

1) Permanent Magnet (PM)
2) Hybrid Stepping Motors

Hybrid Stepping Motors combine the features of the PM Stepping Motors with the features of another type of stepping motor called a Variable Reluctance Motor (VR), which is a low torque and load capacity motor that is typically used in instrumentation. The MicroLYNX System cannot be used with VR motors as they have no permanent magnet.

On Hybrid motors, the phases are wound on toothed segments of the stator assembly. The rotor consists of a permanent magnet with a toothed outer surface which allows precision motion accurate to within ± 3 percent. Hybrid Stepping Motors are available with step angles varying from 0.45° to 15°, with 1.8° being the most commonly used. Torque capacity in hybrid steppers ranges from 5 - 8000 ounce-inches. Because of their smaller step angles, Hybrid motors have a higher degree of suitability in applications where precise load positioning and smooth motion is required.

Sizing a Motor for Your System

The MicroLYNX System contains a bipolar driver which works equally well with both bipolar and unipolar motors (i.e. 8 and 4 lead motors, and 6 lead center tapped motors).

To maintain a given set motor current, the MicroLYNX System chops the voltage using a constant 20kHz chopping frequency and a varying duty cycle. Duty cycles that exceed 50% can cause unstable chopping.

This characteristic is directly related to the motor's winding inductance. In order to avoid this situation, it is necessary to choose a motor with a low winding inductance. The lower the winding inductance, the higher the step rate possible.

Winding Inductance

Since the driver integrated into the MicroLYNX System is a constant current source, it is not necessary to use a motor that is rated at the same voltage as the supply voltage. What is important is that the MicroLYNX System is set to the motor's rated current. See *Section 6: Controlling The Output Current* for more details. As was discussed in the previous section, *Power Supply Requirements,* the higher the voltage used the faster the current can flow through the motor windings. This, in turn, means a higher step rate or motor speed. Care should be taken not to exceed the maximum voltage of the driver. Therefore, in choosing a motor for a system design, the best performance for a specified torque is a motor with the lowest possible winding inductance used in conjunction with highest possible driver voltage. The winding inductance will determine the motor type and wiring configuration best suited for your system. While the equation used to size a motor for your system is quite simple, several factors fall into play at this point. The winding inductance of a motor is rated in milliHenrys (mH) per Phase. The amount of inductance will depend on the wiring configuration of the motor.

The per phase winding inductance specified may be different than the per phase inductance seen by your MicroLYNX System depending on the wiring configuration used. Your calculations must allow for the actual inductance that the driver will see based upon the motor's wiring configuration.

Figure 5.1A shows a stepper motor in a series configuration. In this configuration, the per phase inductance will be 4 times that specified. For example: a stepping motor has a specified per phase inductance of 1.47mH. In this configuration the driver will see 5.88 mH per phase.

Figure 5.1B shows an 8 lead motor wired in parallel. Using this configuration, the per phase inductance seen by the driver will be as specified. Using the following equation, we will show an example of sizing a motor for a MicroLYNX-4 used with an unregulated power supply with a minimum voltage (+V) of 18 VDC:

$$.2 \times 18 = 3.6 \text{ mH}$$

The maximum per phase winding inductance we can use is 3.6 mH.

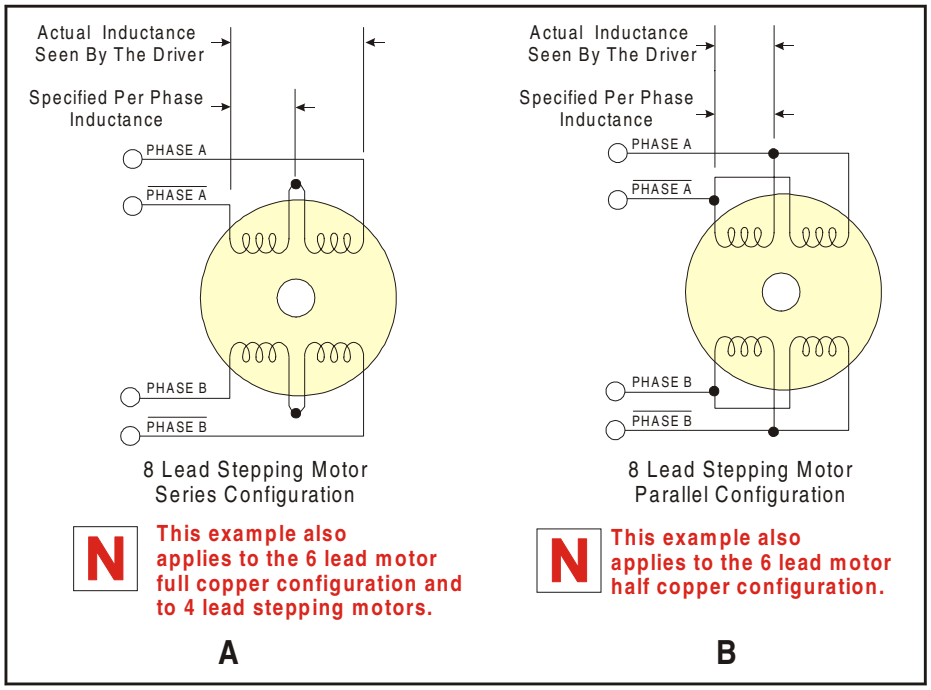


Figure 5.1 A & B: Per Phase Winding Inductance

Maximum Motor Inductance (mH per Phase) =
.2 X Minimum Supply Voltage

NOTE: In calculating the maximum phase inductance, the minimum supply output voltage should be used when using an unregulated supply.

Recommended IMS Motors

IMS stocks the following 1.8° Hybrid Stepping Motors that are recommended for the MicroLYNX System. All IMS motors are CE marked. For more detailed information on these motors, please see the IMS Full Line Catalog or the IMS website at *http//:www.imshome.com/.*

17 Frame (MicroLYNX - 4)

Single Shaft	**Double Shaft**
M2-1713-S	M2-1713-D
M2-1715-S	M2-1715-D
M2-1719-S	M2-1719-D

23 Frame (MicroLYNX - 4/-7)

Single Shaft	**Double Shaft**
M2-2215-S	M2-2215-D
M2-2220-S	M2-2220-D
M2-2232-S	M2-2232-D
M2-2240-S	M2-2240-D

34 Frame (MicroLYNX - 7)

Single Shaft	**Double Shaft**
M2-3424-S	M2-3424-D
M2-3437-S	M2-3437-D
M2-3450-S	M2-3450-D

Enhanced Stepper Motors

IMS also carries a new series of 23 frame enhanced stepping motors that are recommended for use with the IM483H/IM805H. These motors use a unique relationship between the rotor and stator to generate more torque per frame size while ensuring more precise positioning and increased accuracy. The special design allows the motors to provide higher torque than standard stepping motors while maintaining a steadier torque and reducing torque drop-off. The motors are available in 3 stack sizes, single or double shaft, with or without encoders. They handle currents up to 3 Amps in series or 6 Amps parallel, and holding torque ranges from 95 oz.-in. to 230 oz.-in (67 N-cm to 162 N-cm). These CE rated motors are ideal for applications where higher torque is required.

23 Frame High Torque Motors(MicroLYNX - 4/-7)

Single Shaft	**Double Shaft**
MH-2218-S	MH-2218-D
MH-2222-S	MH-2222-D
MH-2231-S	MH-2231-D

IMS Inside Out Stepper Motors

The new Inside Out Stepper (IOS) Motors were designed by IMS to bring versatility to small motors using a unique multi-functional, hollow-core design.

These versatile new motors can be converted to a ball screw linear actuator by mounting a miniature ball screw to the front shaft face. Ball screw linear actuators offer long life, high efficiency and can be field retrofitted. There is no need to throw the motor away due to wear of the nut or screw.
The IOS motors offer the following features:

- The shaft face diameter offers a wide choice of threaded hole patterns for coupling.
- The IOS motor can be direct coupled in applications within the torque range of the motor, eliminating couplings and increasing system efficiency.
- The IOS motor can replace gearboxes in applications where gearboxes are used for inertia dampening between the motor and the load. The induced backlash from the gearbox is eliminated providing improved bi-directional position accuracy.
- Electrical or pnuematic lines can be directed through the center of the motor enabling the motors to be stacked end-to-end or applied in robotic end effector applications. The through hole is stationary preventing cables from being chaffed by a moving hollow shaft.
- Light beams can be directed through the motor for refraction by a mirror or filter wheel mounted on the shaft mounting face.
- The IOS motor is adaptable to valves enabling the valve stem to protrude above the motor frame. The stem can be retrofitted with a dial indicator showing valve position.
- The motor is compatible with IMS bipolar drivers, keeping the system cost low.
- The IOS motor can operate up to 3000 rpm's.

The IOS motor is available in the following frames:

Single Shaft	**IMS P/N**
17 Frame	M3-1713-IOS
23 Frame	M3-2220-IOS
34 Frame	M3-3424-IOS

Motor Wiring

As with the power supply wiring, motor wiring should be run separately from logic wiring to minimize noise coupled onto the logic signals. Motor cabling exceeding 1' in length should be shielded twisted pairs to reduce the transmission of EMI (Electromagnetic Interference) which can lead to rough motor operation and poor system performance overall. For more information on wiring and shielding, please refer to *Rules of Wiring and Shielding* in Section 4 of this manual.

NOTE: The physical direction of the motor with respect to the direction input will depend upon the connection of the motor windings. To switch the direction of the motor with respect to the direction input, switch the wires on either phase A or phase B outputs.

WARNING! Do not connect or disconnect motor or power leads with power applied!

Following are the recommended motor cables:

Dual Twisted Pair Shielded (Separate Shields)

< 5 feet ..... Belden Part# 9402 or equivalent 20 Gauge
> 5 feet ..... Belden Part# 9368 or equivalent 18 Gauge

When using a bipolar motor, the motor must be within 100 feet of the drive.

Connecting the Motor

The motor leads are connected to the following connector pins, which are clearly labeled for ease of use:

Phase	Pin
Phase B\	4
Phase B	3
Phase A\	2
Phase A	1

8 Lead Motors

For the systen designer, 8 lead motors offer a high degree of flexibility in that they may be connected in series or parallel, thus satisfying a wide range of applications.

Series Connection

A series motor configuration would typically be used in applications where a higher torque at low speeds is needed. Because this configuration has the most inductance, the performance will start to degrade at higher speeds. Use the per phase (or unipolar) current rating as the peak output current, or multiply the bipolar current rating by 1.4 to determine the peak output current.

Parallel Connection

An 8 lead motor in a parallel configuration offers more stability but lower torque at lower speeds, but because of the lower inductance there will be higher torque at higher speeds. Multiply the per phase (or unipolar) current rating by 1.96, or the bipolar current rating by 1.4 to determine the peak output current.

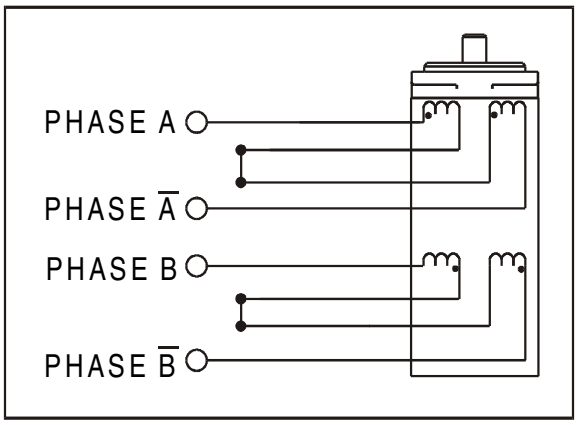


Figure 5.2: 8 Lead Motor, Series Connection

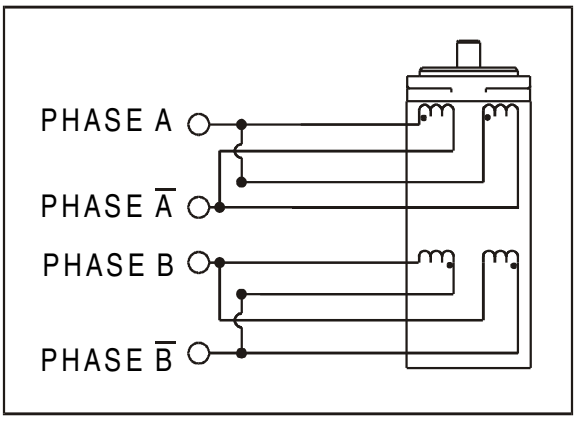


Figure 5.3: 8 Lead Motor, Parallel Connection

6 Lead Motors

As with 8 lead stepping motors, 6 lead motors have two configurations available for high speed or high torque operation. The higher speed configuration, or *half coil, is* so described because it uses one half of the motor's inductor windings. The higher torque configuration, or *full coil,* uses the full windings of the phases.

Half Coil Configuration

As previously stated, the half coil configuration uses 50% of the motor phase windings. This gives lower inductance, hence, lower torque output. As with the parallel connection of 8 lead motor, the torque output will be more stable at higher speeds. This configuration is also referred to as *half copper*. In setting the driver output current, multiply the specified per phase (or unipolar) current rating by 1.4 to determine the peak output current.

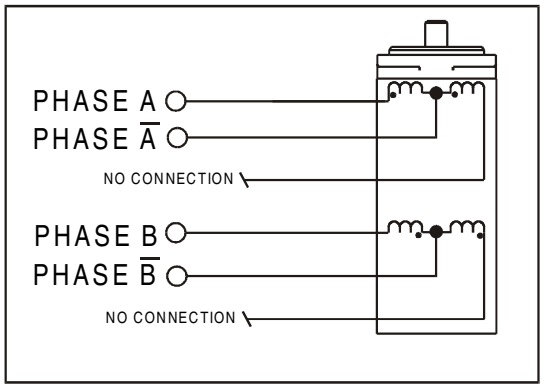


Figure 5.4: 6 Lead Motor, Half Coil Connection

Full Coil Configuration

The full coil configuration on a 6 lead motor should be used in applications where higher torque at lower speeds is desired. This configuration is also referred to as *full copper.* Use the per phase (or unipolar) current rating as the peak output current.

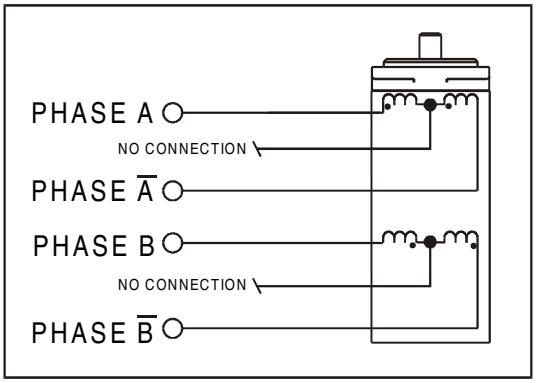


Figure 5.5: 6 Lead Motor, Full Coil Connection

4 Lead Motors

4 lead motors are the least flexible but easiest to wire. Speed and torque will depend on winding inductance. In setting the driver output current, multiply the specified phase current by 1.4 to determine the peak output current.

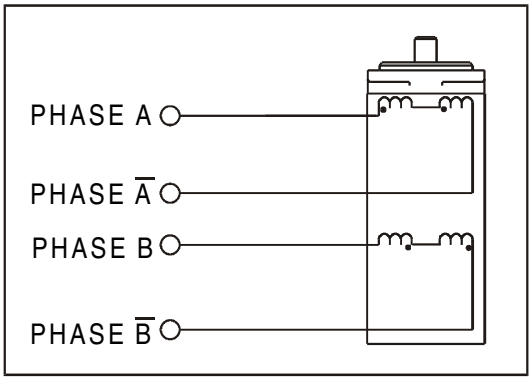


Figure 5.6: 4 Lead Motor

SECTION 6

Controlling the Output Current and Resolution

Section Overview

This section covers the following current control features of the MicroLYNX System:

- Current control variables.
- Determining the output current.
- Setting the output current.
- Setting the motor resolution.

Current Control Variables

One of the unique and powerful features of the MicroLYNX is the precision current control available through the instruction set. Unlike most stepper drives, which only offer the capability of controlling run current and hold current, the MicroLYNX also has the capability of setting the acceleration current. By setting the acceleration current to a higher value, the system designer can deliver more power to the system at the time when it is needed the most: when system inertia must be overcome. Afterwards, when the motor has reached peak velocity, the run current can be set to a lower value, thus reducing motor heating and improving system power efficiency. See *Figure 6.1 and Table 6.1* for the current control variables.

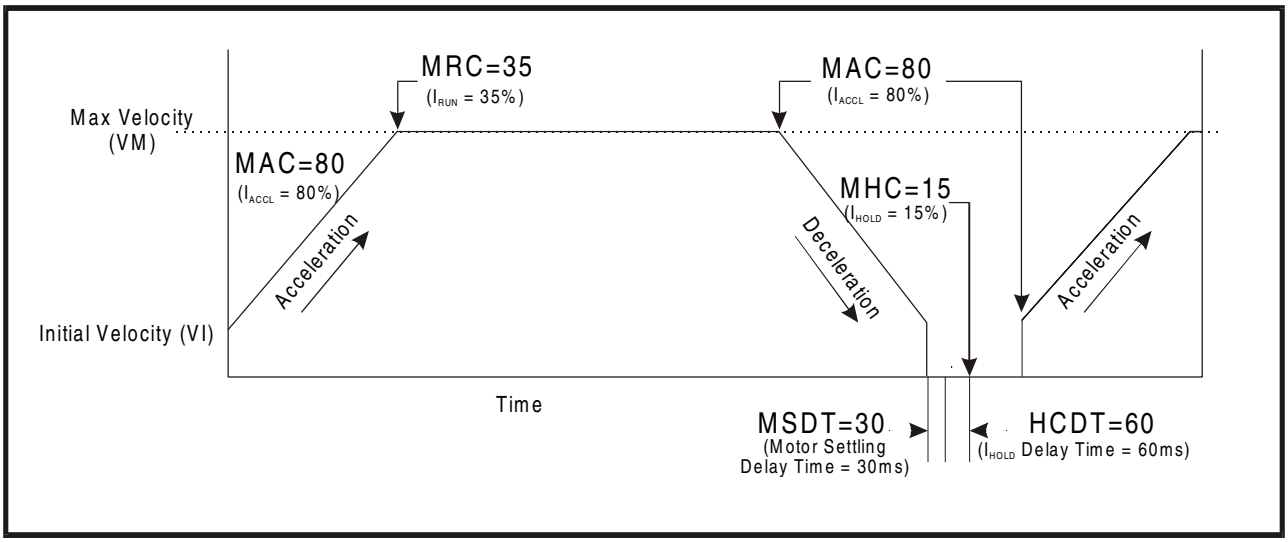


Figure 6.1: Motor Current Control Variables (Values set are for illustration purposes only)

Current Control Variable Summary					
Variable	**Function**	**Usage**	**Unit**	**Range**	**Default Value**
MAC	Motor Acceleration Current Setting	MAC=<num>	Percent	0 - 100	25
MRC	Motor Run Current Setting	MRC=<num>	Percent	0 - 100	25
MHC	Motor Hold Current Setting	MHC=<num>	Percent	0 - 100	5
HCDT	Hold Current Delay Time	HCDT=<num>	Time in milliseconds	0 - 65535	0
MSDT	Motor Settling Delay Time	MSDT=<num>	Time in milliseconds	0 - 65535	0
PMHCC	Position Maintenance Hold Current Change	PMHCC=<num>	Percent	0 - MHC	0

Table 6.1: Motor Current Control Variables

Determining the Output Current

Stepper motors can be configured as 4, 6 or 8 leads. Each configuration requires different currents. Shown below are the different lead configurations and the procedures to determine the peak per phase output current setting that would be used with different motor/lead configurations.

4 Lead Motors

Multiply the specified phase current by 1.4 to determine the peak output current.

Example:
A 4 Lead motor has a specified phase current of 2.0A:

2.0A X 1.4 = 2.8 Amps Peak

6 Lead Motors

A 6 lead motor can be configured two ways: in either the Half Coil Configuration (high speed) or the Full Coil Configuration (higher torque). The current calculation is different for each configuration.

Half Coil Configuration

When configuring a 6 lead motor in the half coil configuration (connected from one end of the coil to the center-tap), multiply the specified per phase (or unipolar) current rating by 1.4 to determine the peak output current.

Example:
A 6 lead motor in half coil configuration has a specified phase current of 3.0A.

3.0A X 1.4 = 4.2 Amps Peak

Full Coil Configuration

When configuring the motor so that full coil is used (connected from end-to-end with the center-tap floating) use the per phase (or unipolar) current rating as the peak output current.

Example:
A 6 lead motor in full coil configuration has a specified phase current of 3.0A.

3.0A per phase = 3.0 Amps Peak

8 Lead Motors

Series Configuration

When configuring the motor windings in series, use the per phase (or unipolar) current rating as the peak output current, or multiply the bipolar current rating by 1.4 to determine the peak output current.

Example #1:
An 8 lead motor in series configuration has a specified unipolar current of 3.0A.

3.0A per phase = 3.0 Amps Peak

Example #2:
An 8 lead motor in series configuration with a specified bipolar current of 2.8A.

2.8 X 1.4 = 3.92 Amps Peak

Parallel Configuration

When configuring the motor windings in parallel, multiply the per phase (or unipolar) current rating by 2.0, or the bipolar current rating by 1.4 to determine the peak output current.

Example #1:
An 8 lead motor in parallel configuration has a specified unipolar current of 2.0A.

2.0A per phase X 2.0 = 4.0 Amps Peak

Example #2:
An 8 lead motor in parallel configuration with a specified bipolar current of 2.8A.

2.8 X 1.4 = 3.92 Amps Peak

Setting the Output Current

The output current on the MicroLYNX is set in software. As previously mentioned, the MicroLYNX differs from other step motor drivers in that the acceleration current can also be set in addition to the run current and holding current.

There are 3 variables in the MicroLYNX instruction set to set these current values:

MAC: Motor Acceleration Current
This value will be used by the MicroLYNX whenever velocity is changing, therefore it will also be the value used when the motor is decelerating.

MRC: Motor Run Current
This value will be used by the MicroLYNX whenever the motor is at peak velocity.

MHC: Motor Holding Current
This value will be used by the MicroLYNX when motion has ceased. The MicroLYNX will change to the hold current setting AFTER the time specified by the MSDT and HCDT variables.

(See *Figure 6.1 and Table 6.1* in the beginning of this section for more detail on these variables and their setup.)

Example Current Setting

For purpose of example we will set the acceleration current to 80%, the run current to 45%, and the holding current to 15%. We will allow the motor 2 seconds to settle into place and delay .5 seconds before reducing the current to the holding value.

```
MAC=80
MRC=45
MHC=15
MSDT=2000
HCDT=500
```

Setting the Motor Resolution

The output resolution of the drive section of the MicroLYNX is set by the MSEL variable. By viewing the table on the right, you can see that there are fourteen (14) resolution settings available with the MicroLYNX. These settings may be changed on-the-fly in either immediate mode or in a program. The operation of this variable is illustrated in the following exercise.

In this excercise we will write a short program that will simply slew the motor and cycle through a few of the binary microstep resolution settings. The lower the resolution is, the higher the speed of the motor.

Enter the following program into the text editor window of the LYNX Terminal:

Microstep Resolution Settings	
MSEL Parameter (Microsteps/Step)	**Microsteps/Rev**
Binary Microstep Resolution Settings (1.8° Motor)	
2	400
4	800
8	1,600
16	3,200
32	6,400
64	12,800
128	25,600
256	51,200

Decimal Microstep Resolution Settings (1.8° Motor)	
5	1,000
10	2,000
25	5,000
50	10,000
125	25,000
250	50,000

Table 7.1: Microstep Resolution Settings

```
MAC=100 'set acceleration current to 75%
MRC=100 'set run current to 75%

PGM 200 'start program at address 200
 SLEW 8000      'slew the motor at 4000 munits/sec
 HOLD 1         'suspend prog. until velocity change
completes
 MSEL=128       'set resolution to 128 msteps/step
 DELAY 1000     'delay program 1 sec.
 MSEL=64        'set resolution to 64 msteps/step
 DELAY 1000     'delay program 1 sec.
 MSEL=32        'set resolution to 32 msteps/step
 DELAY 1000     'delay program 1 sec.
 MSEL=16        'set resolution to 16 msteps/step
 DELAY 1000     'delay program 1 sec.
 MSEL=8         'set resolution to 8 msteps/step
 DELAY 10000
END
PGM
```

Transfer the program to the MicroLYNX by clicking the menu item "Transfer> Download" and selecting "Edit window" as the source. Run the program by typing "EXEC 200" in the terminal. The motor should speed up as it cycles through the resolution setting.

SECTION 7

The Communications Interface

Section Overview

The basic MicroLYNX features two communication interfaces: RS-232 and RS-485. For both channels the BAUD rate is software configured using the BAUD variable to 4800, 9600, 19200 or 38400 bits/sec. The factory default is set to 9600 bits/sec. Default data settings are 8 data bits, 1 stop bit and no parity.

A host computer can be connected to either interface to provide commands to the MicroLYNX System or to multiple MicroLYNX nodes in a system. Since most personal computers are equipped with an RS-232 serial port, it is most common to use the RS-232 interface for communications from the host computer to the MicroLYNX. You will typically want to use this interface option if your Host PC will be within 50 feet of your system. Should your system design place the MicroLYNX at a distance greater than 50 feet, it will be necessary for you to use the RS-485 interface option. You can accomplish this by using either an RS-232 to RS-485 converter, such as the converter sold by IMS (Part # CV-3222), or installing an RS-485 board in an open slot in your host PC.

Covered in detail in this section are:

- RS-232 interface, single MicroLYNX System.
- RS-232 interface, multiple MicroLYNX System.
- RS-485 interface, single MicroLYNX Interface.
- RS-485 interface, multiple MicroLYNX System.
- Connecting and configuring the optional Controller Area Network (CAN) bus.
- MicroLYNX modes of operation.
- MicroLYNX module communication modes.

Connecting the RS-232 Interface

Single MicroLYNX System

In systems with a single MicroLYNX, also referred to as Single Mode, the MicroLYNX is connected directly to a free serial port of the Host PC. Wiring and connection should be performed in accordance with the following table and diagram. In this mode the PARTY ADDRESS switches will be in the OFF position and the PARTY Flag will be set to 0 in software. This is the factory default setting. Please be aware that you cannot communicate with the MicroLYNX in single mode unless those conditions exist.

WARNING! Failure to connect communications ground as shown may result in damage to the Control Module and/or Host!

NOTE! If using the RS-232 Interface Option, the Host PC MUST be less than 50 feet from the Control Module. If your system will be greater than 50 feet from the Host PC you must use the RS-485 interface.

RS-232 MicroLYNX Connection			
MicroLYNX		PC	
10 Pin Header	7 Pin Phoenix	25 Pin Serial Port	9 Pin Serial Port
Pin 3 Receive Data (RX)	Pin 1 Receive Data (RX)	Pin 2 Transmit Data (TX)	Pin 3 Transmit Data (TX)
Pin 2 Transmit Data (TX)	Pin 2 Transmit Data (TX)	Pin 3 Receive Data (RX)	Pin 2 Receive Data (RX)
Pin 5 CGND	Pin 6 CGND	Pin 7 CGND	Pin 5 CGND

Table 7.1: Wiring Connections, RS-232 Interface Single MicroLYNX System

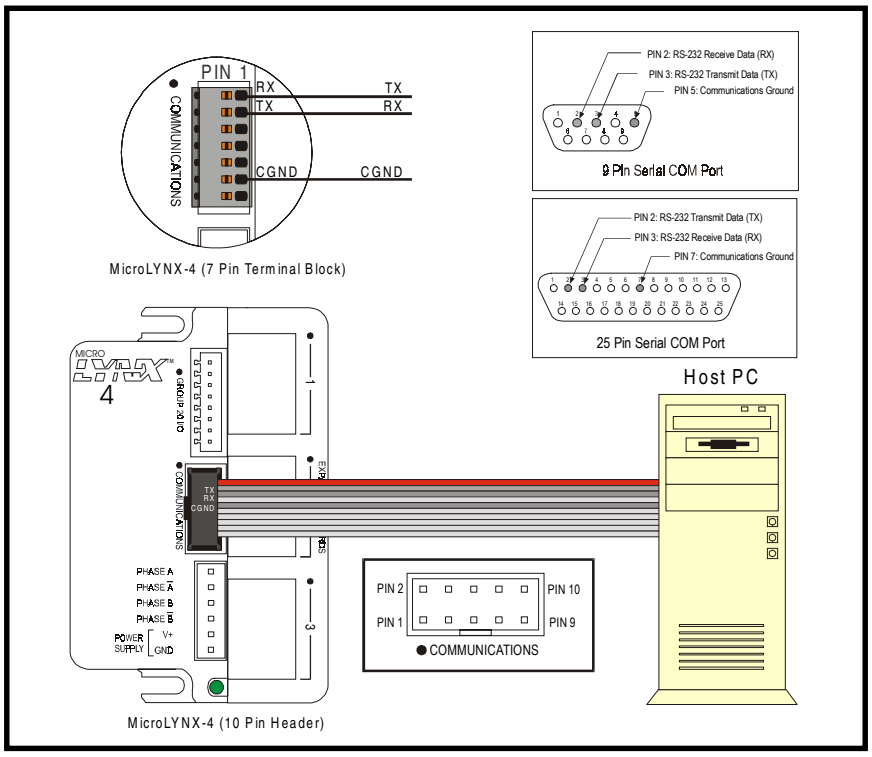


Figure 7.1: Connecting the RS-232 Interface, Single MicroLYNX System

Multiple MicroLYNX Systems

When connecting multiple MicroLYNX nodes in a system using the RS-232 interface it is necessary to establish one MicroLYNX as the HOST. This MicroLYNX will be connected to the Host PC exactly as the system using a single MicroLYNX. The system HOST is established by setting the HOST Flag to True (1) in software. The remaining MicroLYNX nodes in the system must then be connected to the HOST MicroLYNX using the RS-485 interface and will have their HOST Flag set to 0.

In this interface configuration, Host PC communications will be received by the Host MicroLYNX via RS-232 and forwarded to all of the other MicroLYNX nodes in the system via the RS-485 channel. Responses from the individual nodes in the system will be routed back to the Host MicroLYNX via the RS-485 channel, then internally converted to RS-232 before being forwarded back to the Host PC.

In systems with multiple MicroLYNX nodes, it is necessary to communicate with the Host MicroLYNX using PARTY Mode of operation. The MicroLYNX nodes in the system are configured for this mode of operation by setting a unique party address using the address switches, or setting the PARTY Flag to True (1) in software. It is necessary for all of the system nodes to have this configuration selected. When operating in

PARTY Mode, each MicroLYNX in the system will need a unique address to identify it in the system. This can be done using configuration switches A0-A2, or by using the software command SET DN. For example, to set the name of a controller to "A" you would use the following command: SET DN = "A". The factory

Party Mode Address Configuration Switches			
Address	**A2**	**A1**	**A0**
None	OFF	OFF	OFF
A	OFF	OFF	ON
B	OFF	ON	OFF
C	OFF	ON	ON
D	ON	OFF	OFF
E	ON	OFF	ON
F	ON	ON	OFF
G	ON	ON	ON

Table 7.2: Party Mode Address Configuration Switch Settings

default name is "!". To set the address of the controller using the configuration switches use the following table:

The party address switches provide the simplest means for setting up PARTY operation for up to seven (7) MicroLYNX. In setting up your system for PARTY operation via software, the most practical approach would be to observe the following steps:

1. Connect the Host MicroLYNX to the Host PC configured for single mode operation.
2. Establish communications with the HOST MicroLYNX. (For help in doing this, see Software Reference: Using the LYNX Terminal.) Using the Command: SET DN or the configuration switches; give the controller a unique name. If using the software command, this can be any upper or lower case ASCII character or number 0-9. Save the name using the command SAVE.
3. Set the appropriate HOST and PARTY configuration in accordance with the following table and diagram. Remove power.
4. Connect the next MicroLYNX in the system in accordance with the following table and diagram, setting the A0 switch in the ON position.
5. Apply power to the system and establish communications with this module using the name “A”. Rename and save the new name by prefixing the save command with the new name. Remove power.
6. Repeat the last two steps for each additional MicroLYNX in the system.

WARNING! Failure to connect communications ground as shown may result in damage to the Control Module and/or Host!

NOTE! If using the RS-232 Interface Option, the Host PC MUST be less than 50 feet from the Control Module. If your system will be greater than 50 feet from the Host PC you must use the RS-485/RS-485 Interface.

Multiple MicroLYNX System								
Host MicroLYNX			MicroLYNX #1			MicroLYNX #*n*		
10 Pin Header	7 Pin Phoenix	Signal	10 Pin Header	7 Pin Phoenix	Signal	10 Pin Header	7 Pin Phoenix	Signal
6	4	RX+	9	7	TX+	9	7	TX+
7	3	RX-	8	5	TX-	8	5	TX-
8	5	TX-	7	3	RX-	7	3	RX-
9	7	TX+	6	4	RX+	6	4	RX+
10, 5	6	CGND	10, 5	6	CGND	10, 5	6	CGND
Software/Switch Setting			Software/Switch Setting			Software/Switch Setting		
HOST Flag = 1			HOST Flag = 0			HOST Flag = 0		
PARTY Flag = 1 A2, A1, A0 Address Set OR DN=<char>			PARTY Flag = 1 A2, A1, A0 Address Set OR DN=<char>			PARTY Flag = 1 A2, A1, A0 Address Set OR DN=<char>		

Table 7.3: Connections and Settings Multiple MicroLYNX System, RS-232 Interface

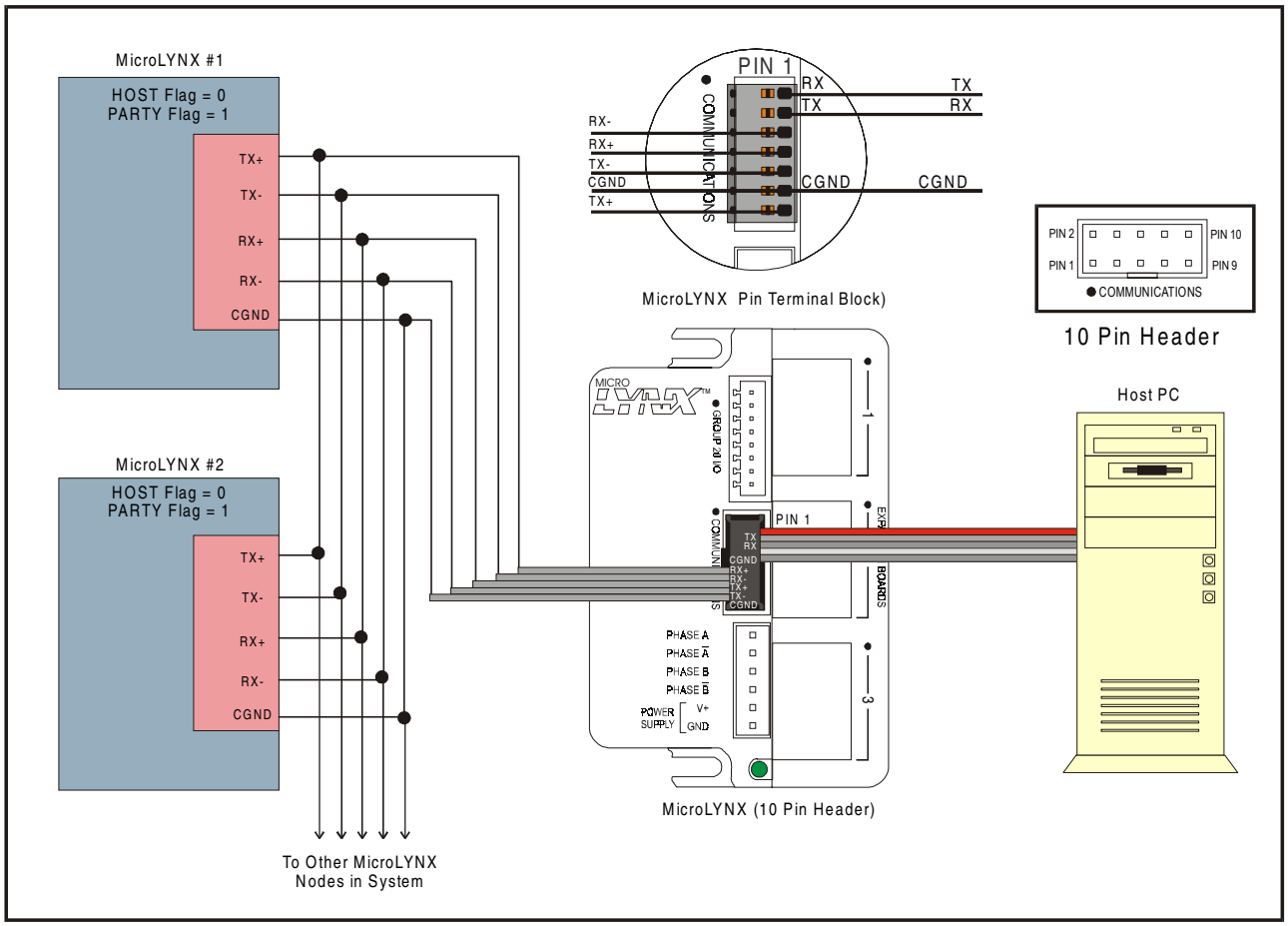


Figure 7.2: RS-232 Interface, Multiple MicroLYNX System

Connecting the RS-485 Interface

Single MicroLYNX System

In a Single Controller System, the RS-485 interface option would be used if the MicroLYNX is located at a distance greater than 50 feet from the Host PC. Since most PC's do not come with an RS-485 board pre-installed, you will have to install an RS-485 board in an open slot in your PC, or purchase an RS-232 to RS-485 converter such as the CV-3222 sold by IMS, to use this connection interface. For wiring and connection information, please use the following table and diagram:

NOTE! The HOST Flag MUST be 0 to communicate with the MicroLYNX System in a Single MicroLYNX System using the RS-485 Interface.

RS-485 Interface Single MicroLYNX System			
RS-232 to RS-485 Converter	**MicroLYNX**		
Signal	**Signal**	**10 Pin Header**	**7 Pin Terminal**
TX-	RX-	7	3
TX+	RX+	6	4
RX-	TX-	8	5
RX+	TX+	9	7
CGND	CGND	10, 5	6

Table 7.4: RS-485 Interface Connections

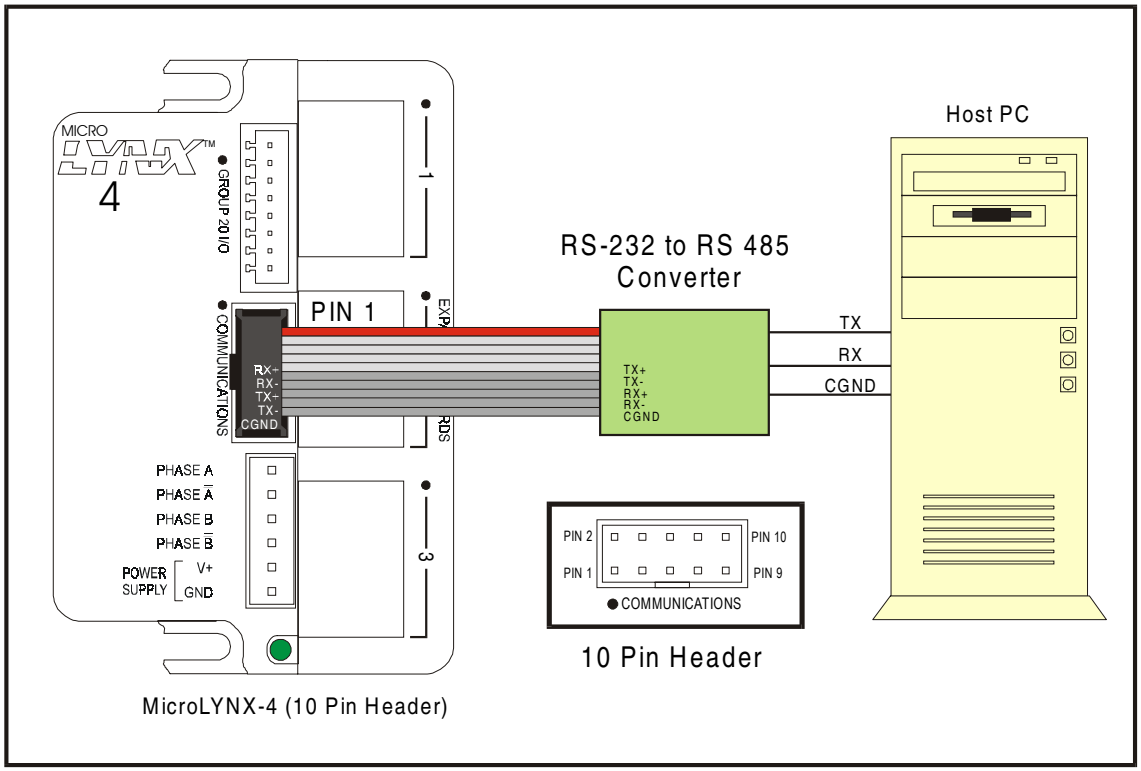


Figure 7.3: RS-485 Interface, Single MicroLYNX System

When using the RS-485 interface in a Multiple MicroLYNX System, the Host PC as well as all of the system nodes to communicate on the RS-485 interface. In this case, there is no Host Interface MicroLYNX so all MicroLYNX nodes in the system should have their HOST flag set to False (0) (Factory Default). The Host PC will be equipped with an RS-485 board or RS-232 to RS-485 converter. In systems with multiple MicroLYNX nodes, it is necessary to communicate with the system nodes using PARTY Mode of operation. The MicroLYNX nodes in the system are configured for this mode of operation by setting the Party Address Switches and setting the PARTY Flag to True (1) in software. It is necessary for all of the nodes in a system

Party Mode Address Configuration Switches			
Address	**A2**	**A1**	**A0**
None	OFF	OFF	OFF
A	OFF	OFF	ON
B	OFF	ON	OFF
C	OFF	ON	ON
D	ON	OFF	OFF
E	ON	OFF	ON
F	ON	ON	OFF
G	ON	ON	ON

Table 7.5: Party Mode Address Configuration Switch Settings

to have this configuration selected. When operating in PARTY Mode, each MicroLYNX node in the system will need a unique address (name) to identify it in the system. This can be done using configuration switches A0-A2, or by using the software command SET DN. For example, to set the name of a controller to "A" you would use the following command: SET DN = "A". The factory default name is "!". To set the address of the controller using the configuration switches use the above table.

Multiple MicroLYNX System RS-485 Interface						
RS-232 to RS-485 Converter or RS-485 Board	**MicroLYNX #1**			**MicroLYNX #*n***		
Signal	**10 Pin Header**	**7 Pin Phoenix**	**Signal**	**10 Pin Header**	**7 Pin Phoenix**	**Signal**
RX+	9	7	TX+	9	7	TX+
RX-	8	5	TX-	8	5	TX-
TX-	7	3	RX-	7	3	RX-
TX+	6	4	RX+	6	4	RX+
CGND	10, 5	6	CGND	10, 5	6	CGND
	Software/Switch Setting			**Software/Switch Setting**		
	HOST Flag = 0			HOST Flag = 0		
	PARTY Flag = 1 A2, A1, A0 Address Set OR DN=<char>			PARTY Flag = 1 A2, A1, A0 Address Set OR DN=<char>		

Table 7.6: RS-485 Interface Connections and Settings, Multiple MicroLYNX System

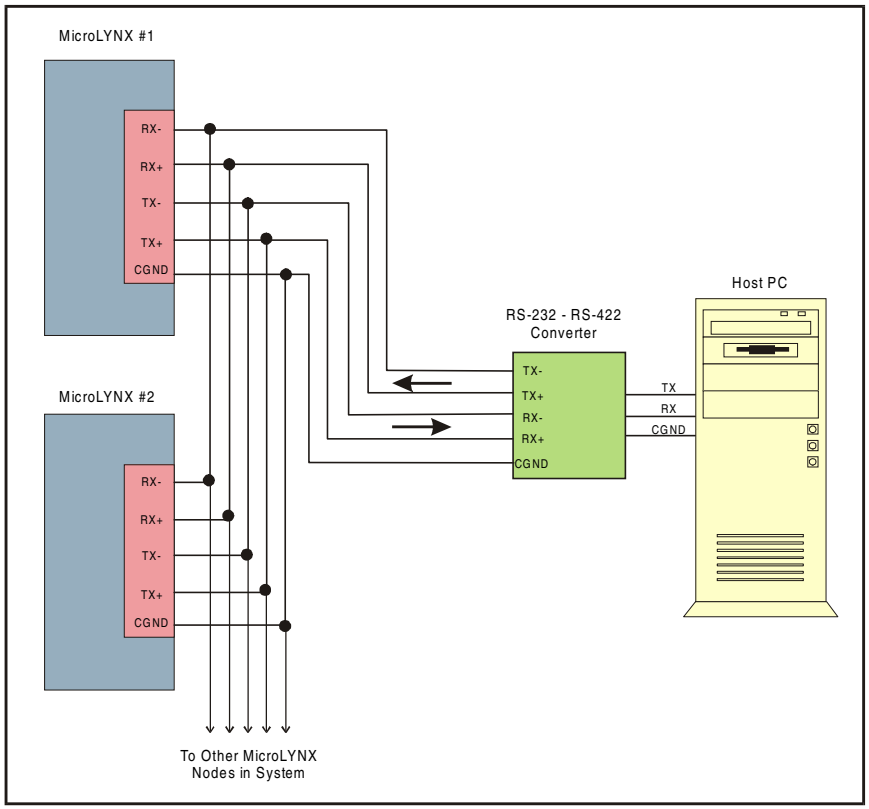


Figure 7.4: RS-485 Interface, Multiple MicroLYNX System

It is also possible to communicate with a MicroLYNX in the system in single mode by sending it a command (with address) to clear the party flag and then communicate with it as in single mode (no line feed terminator) then reset the PARTY Flag when done.

Connecting and Configuring the Optional Controller Area Network (CAN) Bus (IMS P/N MX-CS200-400/-700)

The CAN bus is a high-integrity serial data communications bus for real-time applications originally developed for the automotive industry. Because of its high speed, reliability and robustness, CAN is now being used in many other automation and industrial applications. Using the CAN bus to network controllers, sensors, actuators, etc allows the designer to reduce design time and improve reliability because of readily available components and fewer connections.

A complete discussion of the operation of the CAN bus is beyond the scope of this document.

The MicroLYNX System can be purchased with the capabiltiy to connect to a Controller Area Network (CAN) bus in place of the standard 2 Port RS-232/RS-485 interface. The MicroLYNX with this option conforms to the CAN2.0B Active protocol. CAN2.0B is fully backwards compatible with CAN2.0A, therefore the MicroLYNX can be used on a network with CAN 2.0A devices. There are two receive message frames and one transmit message frame. The CAN version of the MicroLYNX can also be optionally outfitted with RS-232 or RS-485 expansion modules for asynchronous communications.

Connecting to the CAN Bus

To connect to the CAN bus, the only necessary connections are CAN H & CAN L. Since the majority of CAN cabling consists of shielded twisted pair cable, the shield can be connected to the SHIELD connection of the MicroLYNX communications connector. (*See Table 7.7 for pin configuration.*)

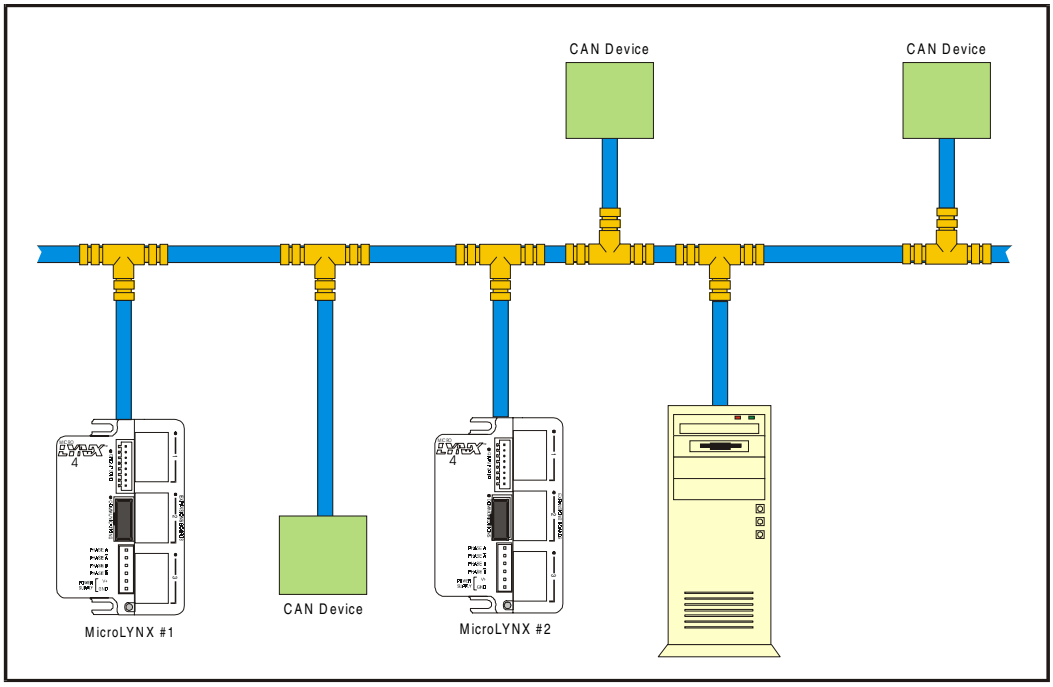


Figure 7.5: Devices on a CAN Bus

Controller Area Network (CAN) Version		
	Connector Option	
Pin #	8 Position Phoenix	10 Pin Header
1	V- (CGND)	N.C.
2	CAN_L	CAN_L
3	SHIELD	V- (CGND)
4	CAN_H	SHIELD
5	N.C. (Reserved for V+)	SHIELD
6	/CONFIG	N.C.
7	N.C.	CAN_H
8		N.C.
9		N.C. (Reserved for V+)
10		/CONFIG

Table 7.7 CAN Pin Configuration

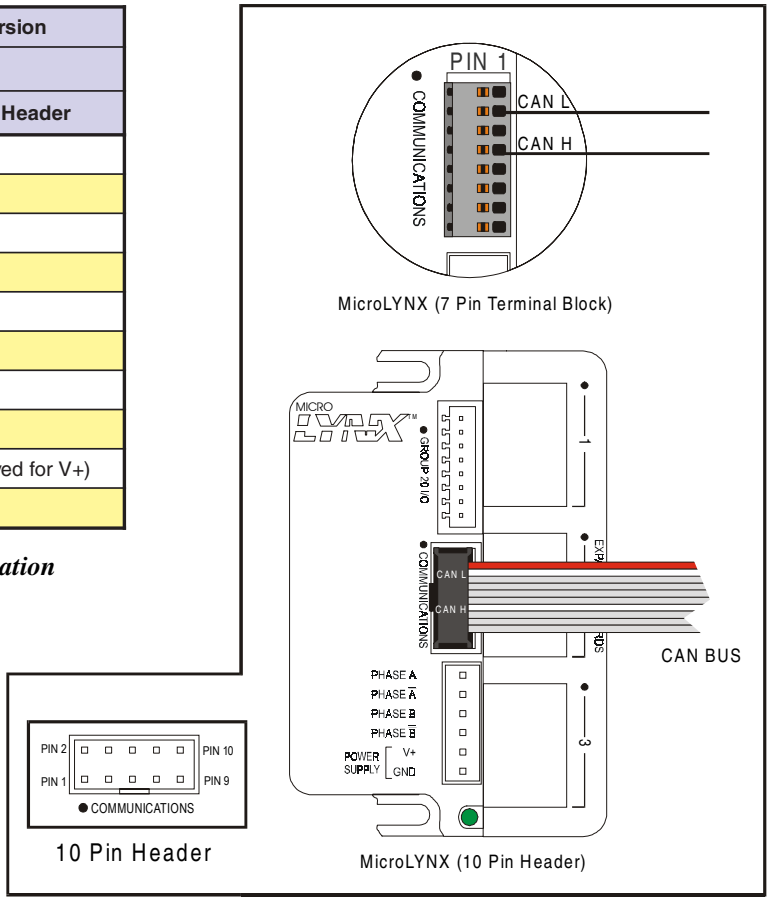


Figure 7.6: Connecting to the CAN Bus

Configuring The CAN Module

The CAN module is placed in configuration mode by holding the CONFIG input LOW on power-up. The module can then be configured using the configuration commands. Care must be taken to ensure proper initialization as no syntax checking is performed on the commands. The CAN module powers up as follows when the config input is held LOW:

BAUD Rate 50kbps
Time Quanta (t_q) before sample point 5
Time Quanta (t_q) after sample point 4
Time Quanta (t_q) before (re) synchronization jump width 2
Identifier ..... Standard 11 bit
Global Mask ..... FFFFh
CAN Receive Identifier FF0h
CAN Transmit Identifier FF2h

When the CONFIG input is held LOW at power-up, a standard message frame (identifier FF2h) will transmit the CAN Module software version number.

The format of all MicroLYNX commands remain unchanged when using the MicroLYNX CAN Module. The CAN Protocol limits the amount of data to be transmitted in a message frame to 8 bytes. Because MicroLYNX commands can be longer than 8 bytes, the MicroLYNX CAN module employs a double buffer scheme. Each enabled receive message frame will buffer a maximum of 64 bytes of data. Once the CAN Module detects a complete MicroLYNX command, the complete command is queued to a 128-byte buffer for transfer to the MicroLYNX.

Any response from the MicroLYNX is queued to a 256-byte buffer and is transferred on the CAN bus when the transmit message frame is enabled. The system designer must be careful not to generate MicroLYNX code that will overflow the 128-byte and 256-byte buffers. All buffers are circular and no checks are made for overflow.

The LYNX Terminal communications software, which ships with the MicroLYNX System, contains a CAN configuration utility to aid in configuring the CAN module. This utility can be accessed via the “setup” menu item on the LYNX Terminal.

CAN Configuration Command Summary

CAN Configuration Command Summary		
Description	**Command / Usage**	**Usage Example**
Initialize CAN Registers	INIT	INIT
Set CAN Bit Timing Registers	BTR0=<hex digit><hex digit> BTR1=<hex digit><hex digit>	BTR0=49 BTR1=34
Set Global Mask Registers	GMS0=<hex digit><hex digit> GMS1=<hex digit><hex digit> UGML0=<hex digit><hex digit> UGML1=<hex digit><hex digit> LGML0=<hex digit><hex digit> LGML1=<hex digit><hex digit>	GMS0=FF GMS1=FF UGML0=FF UGML1=FF LGML0=FF LGML1=F8
Setup Message Frames	FRM=<frame #><valid flag><extended ID flag>	FRM=210
Set Message Frame Arbitration Registers	UAR0=<frame#><hex digit><hex digit> UAR1=<frame#><hex digit><hex digit> LAR0=<frame#><hex digit><hex digit> LAR1=<frame#><hex digit><hex digit>	UAR0=2A3 UAR1=200 LAR0=200 LAR1=200
MicroLYNX Mode	LMODE=<flag>	LMODE=0
MicroLYNX Party Address	LADDR=<address>	LADDR=X
MicroLYNX Prompt	LPRMPT=<char>	LPRMPT=>
MicroLYNX BAUD Rate	LBAUD=<baud#><baud#>	LBAUD=38

Table 7.8: CAN Configuration Command Summary

To Initialize the CAN Module.

Command : **INIT**

The factory default settings for the CAN module are detailed below:

BAUD Rate	50kbps
Time Quanta (t_q) before sample point	5
Time Quanta (t_q) after sample point	4
Time Quanta (t_q) before (re) synchronization jump width	2
Global Mask	FFFFh
CAN Receive Identifier	FF0h
CAN Transmit Identifier	FF2h
BTR0	53h
BTR1	34h
GMS0	FFh
GMS1	FFh
UGML0	FFh
UGML1	FFh
LGML0	FFh
LGML1	F8h
UMLM0	FFh
UMLM1	FFh
LMLM0	FFH
LMLM1	FFh
Message Frame 1	not valid
Message Frame 2	not valid
Message Frame 3	not valid
MicroLYNX Mode	single
MicroLYNX Party Address	!
MicroLYNX Prompt	>
MicroLYNX BAUD Rate	9600

The use of the “INIT” instruction will restore these defaults in the CAN module. There are several new enhancements to the LYNX instrutcion set which add the functions of the CANmodule while maintaining backward compatibility with the modular LYNX system. The following instructions and variables are specific to the CAN Module. These are introduced here and covered in more detail in the Software Reference. *Table 6.8* contains a summary of the configuration commands for the CAN Module.

The MicroLYNX must be in ASCII communications mode for use with the CAN Module.

To Set the CAN Bit Timing Registers

Command:
BTR0=<hex digit><hex digit>
BTR1=<hex digit><hex digit>

Usage Example:
BTR0=49
BTR1=34

This sets 100 Kbaud, 5 time quanta before the sample point, 4 time quanta after the sample point, and 2 time quanta for (re)synchronization jump width.

CAN Bit Timing Registers								
	7	6	5	4	3	2	1	0
BTR0	SJW		BRP					
BTR1	0	TSEG2			TSEG1			

Table 7.9: CAN Bit Timing Registers

SJW - (Re)synchronization jump width. Adjust the bit time by (SJW+1) time quanta for resynchronization. Valid values are 0 - 3.
BRP – Baud rate prescalar. The oscillator frequency (10 MHz) is divided by (BRP+1) to generate the bit time quanta.

TSEG2 – Time segment after sample point. There are (TSEG2+1) time quanta after the sample point. Valid values are 1 - 7.
TSEG1 – Time segment before sample point. There are (TSEG1+1) time quanta before the sample point. Valid values are 2 - 15.

Sample Bit Timing Register Settings		
Default Time Quanta Settings 5 time quanta before sample point. 4 time quanta after sample point. 2 time quanta for (re)synchronization jump width.		
BAUD (kbps)	**BTR0**	**BTR1**
50	53h	34h
100	49h	34h
250	43h	34h
500	41h	34h
1000	40h	34h

Table 7.10: Sample Bit Timing Register

A bit time is subdivided into four segments as defined in the CAN specification. Each segment is a multiple of the time quantum (t_q). The synchronization segment (Sync-Seg) is always 1 t_q long. The time before the sample point is defined as the combined times of propagation time segment and the phase buffer segment1 (TSEG1). The time after the sample point is defined as the phase buffer segment2 (TSEG2).

The Bit Timing Registers can also be set using the Bit Rate/Bit Timing Calculator utility in the LYNX Terminal software that comes with the MicroLYNX *(see Figure 7.7)*. This utility can be accessed from the Setup > Configure CAN menu item on the main LYNX terminal window.

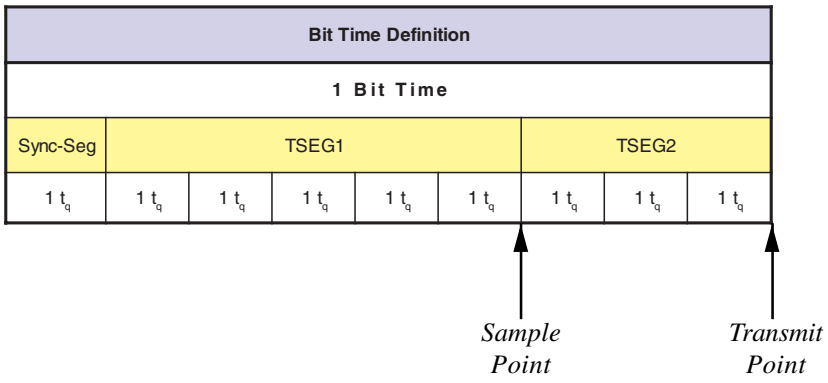


Table 7.11: CAN Bit Time Definition

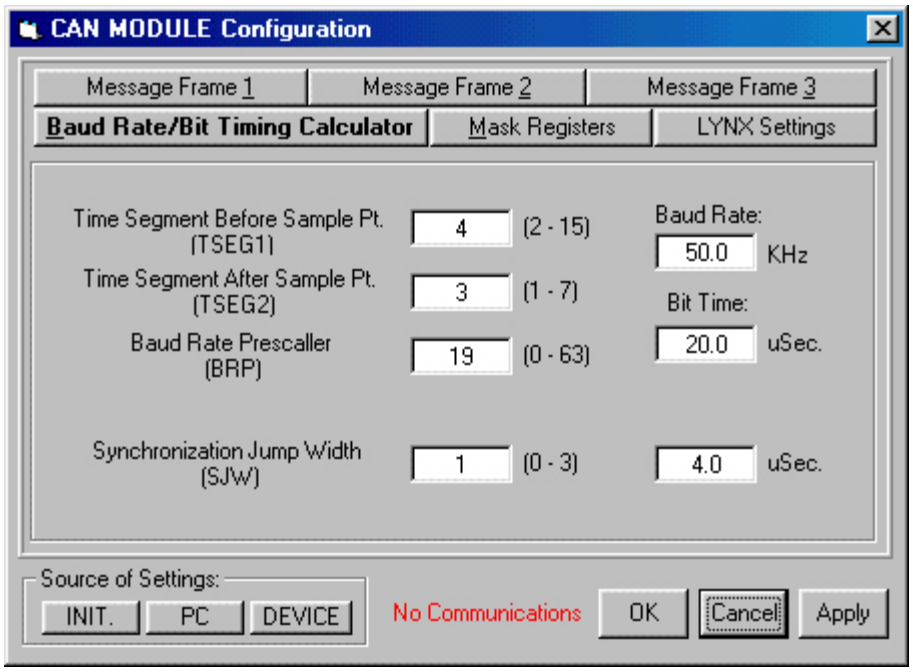


Figure 7.7: Bit Register Configuration Dialog from LYNX Terminal

For installation instructions for LYNX Terminal see *The Software Reference: The LYNX Terminal Software* on the LYNX CD.

To Set The Global Mask Registers

Command:	Usage Example:
GMS0=<hex digit><hex digit>	**GMS0=FF**
GMS1=<hex digit><hex digit>	**GMS1=FF**
UGML0=<hex digit><hex digit>	**UGML0=FF**
UGML1=<hex digit><hex digit>	**UGML1=FF**
LGML0=<hex digit><hex digit>	**LGML0=FF**
LGML1=<hex digit><hex digit>	**LGML1=F8**

SID28-18 – Standard Identifier (11-bit)
EID28-0 – Extended Identifier (29-bit)

Incoming message frames are masked with the appropriate global mask. If the bit position in the global mask register is 0 (don't care), then the bit position will not be compared with the incoming message's identifier.

Global Mask Registers								
	7	6	5	4	3	2	1	0
GMS0	SID 28-21							
GMS1	SID 20-18				1	1	1	1
UGML0	EID 28-21							
UGML1	EID 20-13							
LGML0	EID 12 -5							
LGML1	EID 4-0					0	0	0

Table 7.12: Global Mask Registers

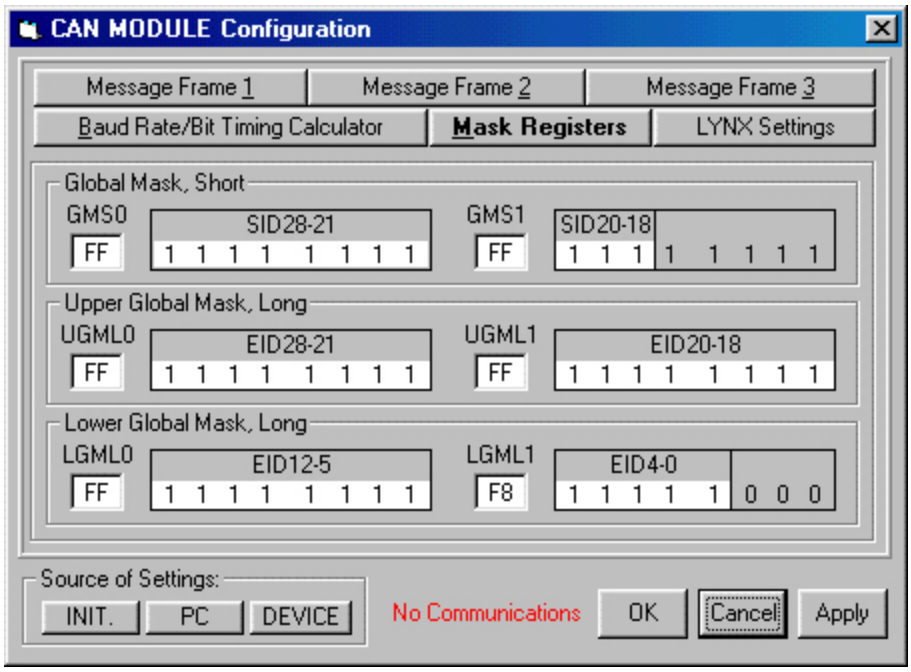


Figure 7.8: Setup Dialog for Global Mask Registers in LYNX Terminal

To Setup Message Frames

Command: **FRM=<frame#><valid flag><extended ID flag>**

<frame#>	= frame number (1-3). Frames 1 and 2 are fixed as receive frames. Frame 3 is fixed as a transmit frame.
<valid flag>	= 1 frame valid
<valid flag>	= 0 frame not valid
<extended ID flag>	= 1 extended identifier
<extended ID flag>	= 0 standard identifier

The CAN module will only operate on valid message objects.

Example:
FRM=210
This sets message 2 valid, using the standard identifier.

Set Message Frame Arbitration Registers

Command:	Usage Example:
UAR0=< frame#><hex digit><hex digit>	**UAR0=2A3**
UAR1=< frame#><hex digit><hex digit>	**UAR1=200**
LAR0=< frame#><hex digit><hex digit>	**LAR0=200**
LAR1=< frame#><hex digit><hex digit>	**LAR1=200**
< frame#> = frame number	(1-3)This sets message 2 arbitration registers to A30h.

ID28-18 – Identifier of a standard message. ID17-0 set to 0 for a standard message.
ID28-0 – Identifier of an extended message.

The arbitration registers are used for acceptance filtering of incoming messages and to define the identifier of outgoing messages. There must not be more than one valid message object with a particular identifier at any time. If some bits are masked by the global mask registers, then the identifiers of the valid message objects must differ in the remaining bits which are used for acceptance filtering.

This sets message 2 arbitration registers to A30h.

Message Frame Arbitration Registers								
	7	6	5	4	3	2	1	0
UAR0	ID28 - 21							
UAR1	ID20 - 18			ID17 - 13				
LAR0	ID12 - 5							
LAR1	ID4 - 0					0	0	0

Table 7.13: Message Frame Arbitration Registers

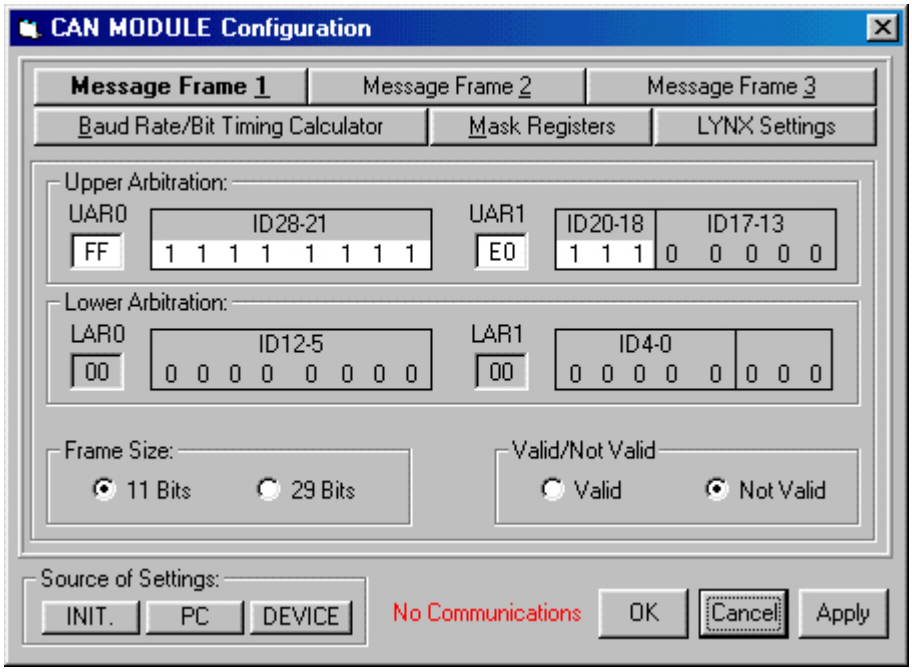


Figure 7.9: Message Frame Setup Dialog from LYNX Terminal

Defining the MicroLYNX Mode (Single or Party)

This command identifies to the CAN module the MicroLYNX mode.

Command:
LMODE=<flag>
<flag> = 0 single mode
<flag> = 1 party mode

Example:
LMODE=0
This indicates to the CAN module that the MicroLYNX is operating in single mode.

Setting the MicroLYNX Party Address

Command:
LADDR=<address>

<address> = any valid MicroLYNX address.

Usage Example:
LADDR=X

This indicates to the CAN module that the MicroLYNX party address is X.

This command identifies to the CAN module the MicroLYNX address when party mode is enabled in the MicroLYNX.

MicroLYNX Prompt

Command:
LPRMPT=<char>

<char> = any valid MicroLYNX prompt character.

Usage Example:
LPRMPT>
This indicates to the CAN module that the MicroLYNX prompt is the > character.

This command identifies to the CAN module the MicroLYNX prompt character.

MicroLYNX Baud Rate

Command:
LBAUD=<baud#><baud#>

<baud#><baud#> = 48 4800 baud
<baud#><baud#> = 96 9600 baud
<baud#><baud#> = 19 19200 baud
<baud#><baud#> = 38 38400 baud

Usage Example:
LBAUD=38
This indicates to the CAN module that the MicroLYNX baud rate is 38400 baud.

This command identifies to the CAN module the MicroLYNX baud rate.

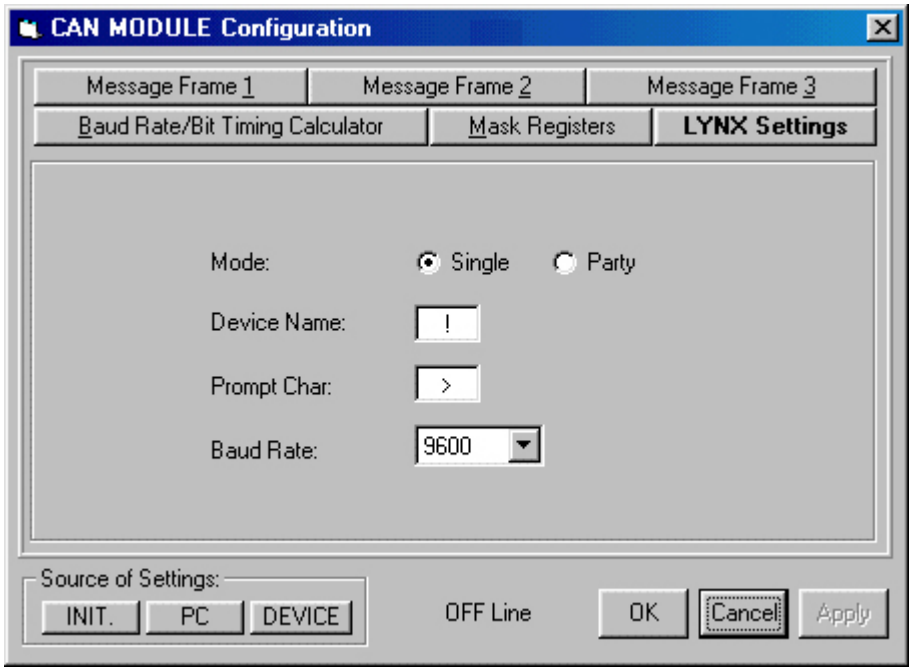


Figure 7.10: LYNX Setup Dialog from LYNX Terminal

MicroLYNX Modes of Operation

There are three modes of operation for the MicroLYNX. These are Immediate Mode, Program Mode and EXEC Mode.

Immediate Mode

In this mode the MicroLYNX responds to instructions from the user that may be a result of the user typing instructions directly into a host terminal, or of a user program running on the host which communicates with the MicroLYNX.

Program Mode

The second mode of operation of the MicroLYNX is Program Mode. All user programs are entered in this mode. Unlike the other modes of operation, no commands or instructions can be issued to the MicroLYNX in Immediate Mode. This mode is exclusively for entering programs for the MicroLYNX. The command to enter Program Mode is PGM <address>. When starting Program Mode, you must specify at what address to enter the program instructions in the program space. Simply type PGM again when you have finished entering your program commands to go back to Immediate Mode.

EXEC Mode

In EXEC Mode a program is executed either in response to the EXEC instruction from the user in Immediate Mode, or in response to a specified input. While the MicroLYNX is running a program, the user may still communicate with it in Immediate Mode. As part of a user program, the MicroLYNX may start a second task using the RUN instruction. Thus, there can be two tasks running on the MicroLYNX at the same time, a foreground task (started by the EXEC instruction in Immediate Mode) and a background task (started by the RUN instruction in Immediate Mode or EXEC Mode).

MicroLYNX Communication Modes

When the MicroLYNX is operating in Immediate Mode, there are two methods of communicating. The first is ASCII where the instructions are communicated to the MicroLYNX in the form of ASCII mnemonics and data is also given in ASCII format. The second is binary where the instruction is in the form of an OpCode and numeric data is given in IEEE floating point hex format. In binary mode, there is also the option of including a checksum to ensure that information is received properly at the MicroLYNX. The BIO flag controls the method of communication. When it is True (1) the binary method should be used, and when it is False (0) the ASCII method should be used.

ASCII

ASCII is the most common mode of communicating with the MicroLYNX System. It allows the use of readily available terminal programs such as HyperTerminal, ProComm and the new LYNX Terminal.

When using the ASCII method of communications, the MicroLYNX tests for four special characters each time a character is received. These characters are given in the following table along with an explanation of what occurs when the character is received.

The command format in ASCII mode when the MicroLYNX is in Single Mode (PARTY = FALSE) is:

<Mnemonic><white space><ASCII data for 1st parameter>, <ASCII data for 2nd parameter>, … , <ASCII data for nth parameter><CR/LF>

The mnemonics for MicroLYNX instructions, variables, flags and keywords are given in Section 16 of this document. White space is at least one space or tab character. CR/LF represent the carriage return line feed

characters that are transmitted in response to the Enter key on the keyboard provided the ASCII setup specifies “Send line feeds with line ends”. Note that there need not be a space between the data for the last parameter and the CR/LF. Also note that if there is only one parameter, the CR/LF would immediately follow the data for that parameter.

The command format in ASCII mode, when the MicroLYNX is in Party Mode (PARTY = TRUE), would be identical to that in Single Mode with the exception that the entire command would be preceded by the MicroLYNX’s address character (stored in DN) and terminated by a CTRL-J rather than ENTER:

<Address character><Mnemonic><white space><ASCII data for 1st parameter>, <ASCII data for 2nd parameter>, … , <ASCII data for nth parameter><CTRL-J>

ASCII Mode Special Command Characters	
Character	**Action at MicroLYNX**
<esc> Escape Key	Terminates all active operations and all running programs.
<^C> Ctrl+C Keys	Terminates all active operations and all running programs, forces a reset of the MicroLYNX.
<BKSP> Backspace Key	Moves the cursor back one in the buffer to correct a typing error.
<CR> or <LF> Carriage Return or Line Feed	Depending on the mode, either Single or Party. <LF> is not necessary in Single Mode communications. <CTRL+J> is the same as <LF> (0A Hex)

Table 7.14: ASCII Mode Special Command Characters

Binary

Binary mode communications is faster than ASCII and would most likely be used in a system design where the communication speed is critical to system operation. This mode cannot be used with standard terminal software.

The command format in binary mode when the MicroLYNX is in Single Mode (PARTY = FALSE) is:

<20H><character count><opcode><Field type for 1st parameter><IEEE hex data for 1st parameter><0EH><Field type for 2nd parameter><IEEE hex data for 2nd parameter><0EH> … <Field type for nth parameter><IEEE hex data for nth parameter><optional checksum>

Binary Hex Codes	
Hex Code	**Data Type**
01	Label Text
02	ASCII Text
03	1 byte unsigned
04	2 byte signed
05	2 byte unsigned
06	4 byte signed
07	4 byte unsigned
08	4 byte float

Table 7.15: Binary Hex Codes

Note that <20H> is 20 hex, the character count is the number of characters to follow the character count not including the checksum if one is being used. The OpCodes for MicroLYNX instructions, variables, flags and keywords are given in Sections 15 and 16 of this document. The Field type byte will be one of the following based on the type of data that is expected for the specific parameter:

<0EH> is 0E hex, which is a separator character in this mode. Finally, the optional checksum will be included if CSE is TRUE and excluded if it is FALSE. If included, the checksum is the low eight bits of the complemented sixteen-bit sum of the address field (20H here), character count, OpCode, all data fields and separators (0E hex).

SECTION 8

Configuring the Isolated Digital I/O

Section Overview

This section covers the usage of the Isolated Digital I/O which isavailable on the MicroLYNX System

- Electrical Characteristics.
- The Isolated Digital I/O:
  - Configuring an Input
  - Setting the Digital Input Filtering for the Isolated I/O
  - Configuring an Output
  - Setting the Binary State of an I/O Group

Electrical Characteristics

Number of I/O	6
Input Voltage	+5 to +24VDC
Output Current Sink	350mA
Input Filter Range	215Hz to 21.5kHz (Programmable)
Pull-ups	7.5kOhm individually switchable
Pull-up Voltage	
Internal	+5VDC
External	+24 VDC
Protection	Over temp, short circuit, inductive clamp
Isolated Ground	Common to the 6 I/O

The Isolated Digital I/O

The MicroLYNX System comes standard with a set of six (6) +5 to +24VDC I/O lines which may be programmed individually as either general purpose or dedicated inputs or outputs, or collectively as a group. The isolated digital I/O may also be expanded to twenty-four (24) lines in groups of six (6).

The I/O groups and lines are numbered in the following fashion:

Group 20 = Lines 21 - 26 (Standard)
Group 30 = Lines 31 - 36 (Optional)
Group 40 = Lines 41 - 46 (Optional)
Group 50 = Lines 51 - 56 (Optional)

The isolated digital I/O may be defined as either active HIGH or active LOW. When the I/O is configured as active HIGH, the level is +5 to +24 VDC and the state will be read as a “1”. If the level is 0 VDC then the state will be read as “0”. Inversely, If configured as active LOW, then the state of the I/O will be read as a “1” when the level is LOW, and a “0” when the level is HIGH. The active HIGH/LOW state is configured by the third parameter of the IOS variable, which is explained further on. The goal of this I/O configuration scheme is to maximize compatibility between the MicroLYNX and standard sensors and switches. TheMicroLYNX I/O scheme is a powerful tool for machine and process control. Because of this power, a level of complexity in setup and use is found that doesn’t exist in controllers with a less capable I/O set.

Uses of the Isolated Digital I/O

The isolated I/O may be utilized to receive input from external devices such as sensors, switches or PLC outputs. When configured as outputs, devices such as relays, solenoids, LED's and PLC inputs may be controlled from the MicroLYNX. Depending on the device connected, the input or output may be pulled-up to either the internal +5VDC supply or an external +5 to +24VDC supply, or the I/O lines may be pulled-down to ground.
These features, combined with the programmability and robust construction of the MicroLYNX I/O open an endless vista of possible uses for the I/O in your application.

Each I/O line may be individually programmed to any one of 8 dedicated input functions, 7 dedicated output functions, or as general purpose inputs or outputs. The I/O may be addressed individually, or as a group. The active state of the line or group may also be set. All of these possible functions are accomplished with of the IOS variable

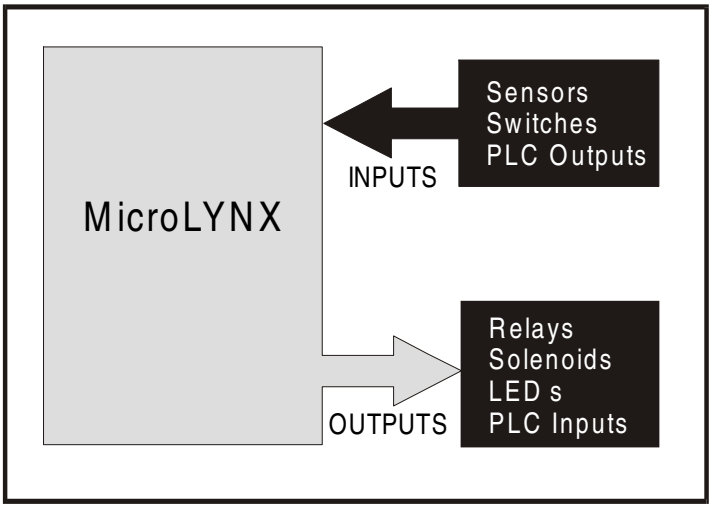


Figure 8.1: Isolated I/O Applications

The IOS Variable

The IOS variable has three parameters when used to configure the isolated digital I/O. These are:

1] **I/O Line Type:** Specifies the the type of I/O that the line or group will be configured as, i.e. general purpose or dedicated function.
2] **I/O Line Function:** Either an input or an output.
3] **Active State:** Specifies whether or not the line will be active HIGH or active LOW.

The default configuration of the standard I/O set is: 0,0,1. This means that by default each line in group 20 is configured to be a General Purpose (0), Input (0), which is active when HIGH (1). Table 8.1 on the following page and the exercises illustrate possible configurations of the IOS.

NOTE: When configuring a dedicated input or output, the second parameter of the IOS Variable MUST match the function, either input or output, or an error will occur.

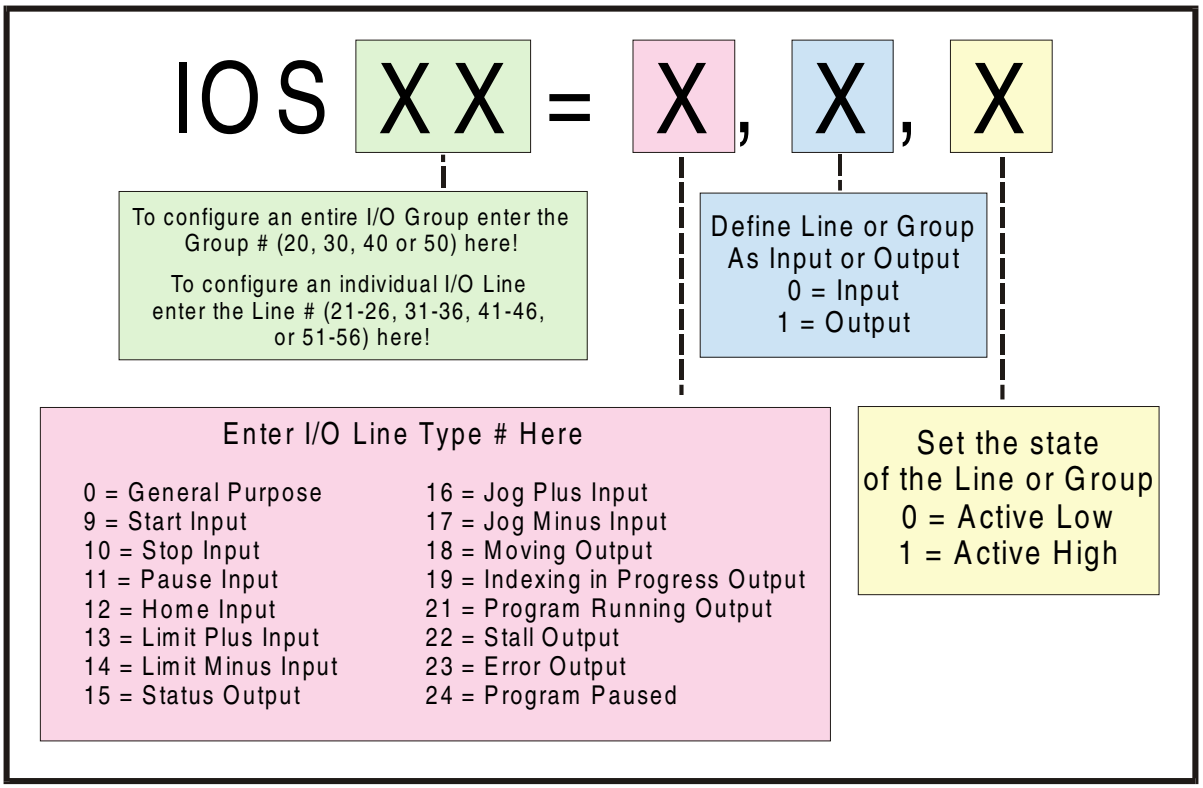


Table 8.1: IOS Variable Settings

Configuring an Input

Figure 8.2 below illustrates the Input Equivalent Circuit of the Isolated I/O being used with a switch. To illustrate the usage of an input you will go through the steps to configure this switch to start a simple program at Line 1000 to index a motor 200 user units. First you must configure the I/O Line 21 as a "GO" input:

```
IOS 21 = 9, 0, 0
```

To break this command down:

IOS 21 - Identifies the I/O Line we are setting as 21.

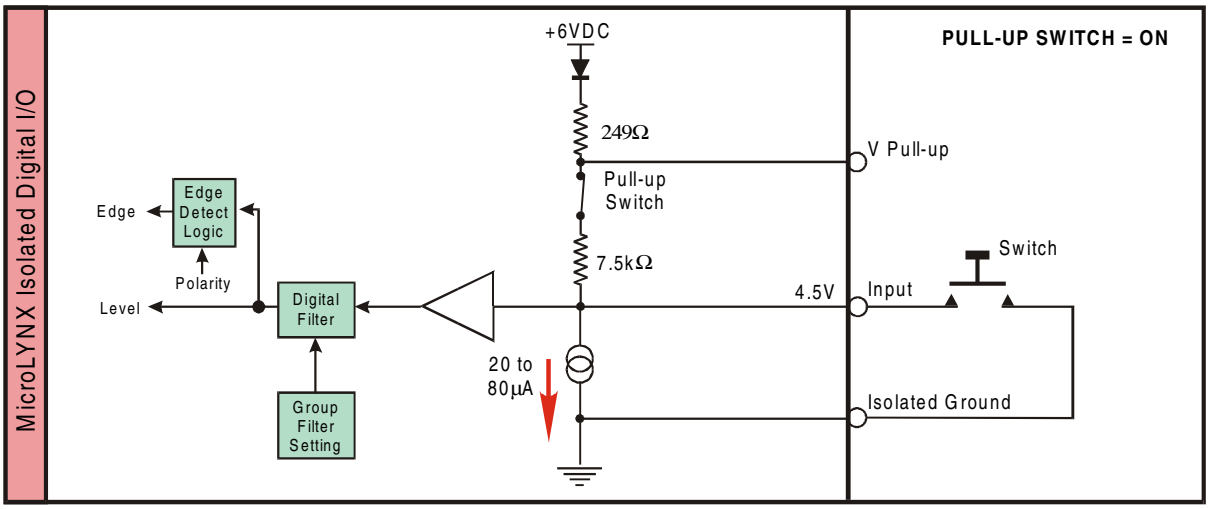


Figure 8.2: Isolated I/O Input Equivalent Circuit

9 - Configures the I/O Type to “GO”.
0 - Configures I/O as Input.
0 - Configures I/O as Active LOW.

When the Input Type “GO” is selected it will always look to execute a program located at line 1 of program memory. Therefore, to run a program at line 1000 you must do the following:

```
PGM 1        'Records program at line 1 of memory space
EXEC 1000    'Execute program located at line 1000 of memory space
END          'Terminates Program
PGM          'Switches system back to immediate mode

PGM 1000     'Records program at line 1000 of memory space
MOVR 200     'Move relative to current position 200 user units
HOLD 2       'Hold program execution until specified motion is
             'completed
END
PGM
```

Configuring the Digital Filtering

User definable Digital filtering makes the LYNX well suited for noisy industrial environments. The filter setting is software selectable using the *IOF Variable* with a minimum guaranteed detectable pulse width of 18 microseconds to 2.3 milliseconds.

Table 8.2 illustrates the IOF settings.

The filter setting will reject any frequency above the specified bandwidth. For example:

```
IOF 2 = 3    'Set the
Digital Filter for I/O
Group 20 to 3.44kHz
```

IOF Filter Settings for the General Purpose Isolated I/O IOF=<num> (<num> = 0-7)		
Filter Setting	**Cutoff Frequency**	**Minimum Detectable Pulse Width**
0	27.5 kHz	18 microseconds
1	13.7 kHz	36 microseconds
2	6.89 kHz	73 microseconds
3	3.44 kHz	145 microseconds
4	1.72 kHz	290 microseconds
5	860 Hz	581 microseconds
6	430 Hz	1.162 milliseconds
7 (default)	215 Hz	2.323 milliseconds

Table 8.2: Digital Filter Settings for the Isolated I/O

This setting will cause any signal above 3.44 kHz on I/O lines 21-26 to be rejected. The default filter setting for the isolated I/O groups is 7, or 215Hz.

Configuring an Output

Figure 8.3 illustrates the Output equivalent circuit of the Isolated I/O. When used as an output the I/O line is able to sink 350mA continuous for each output, or a total of 1.5A for the entire I/O Group. See Section 9: *The Isolated I/O Module* for detailed specifications. In the usage example we will use an LED on I/O Line 31 for the load. We will use the same program from the input example, only we will use the output to light the LED while the motor is moving.

```
IOS 31 = 18, 1,  1
```

Using Table 8.1 on the previous page we can break this setting down as follows:
IOS 31 - Identifies that I/O line 31 is being configured.
18 - Configures the I/O Type as “Moving”.
1 - Configures the I/O line as an output.
1 - Configures the Line as “Active HIGH”.

Now when the input program above is executed, the LED will be lit during the move.

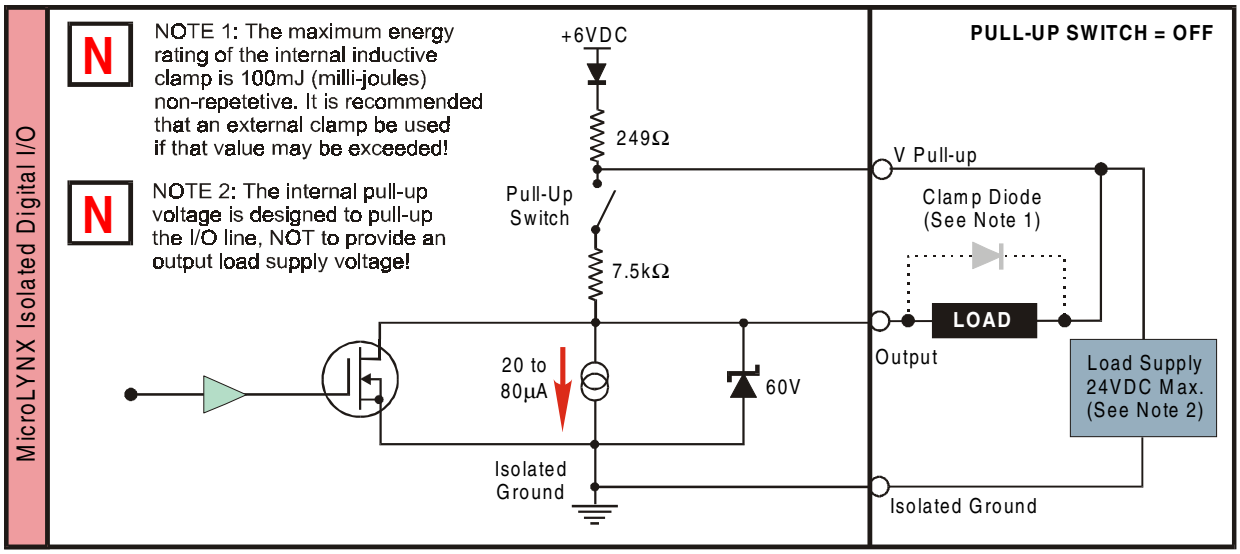


Figure 8.3: Isolated I/O Output

The IO Variable

After configuring the I/O by means of the IOS variable, we need to be able to do two things with the I/O.

1] Write to an output, or group of outputs, thus setting or changing its (their) state.

2] Read the states of either inputs or outputs. We can use this information to either display those states to our terminal, or to set up conditions for branches and subroutine calls within a program.

We can also use this command to write or read the state of an entire I/O group.

Read/Write a Single I/O Line

To read the state of a single input or output, the following would be typed into the terminal:

```
PRINT IO 21
```

The response from this would be 1 or 0, depending on the state of the line.
The state of an input or output in a program can be used to direct events within a LYNX program by either calling up a subroutine using the "CALL" instruction, or conditionally branching to another program address using the "BR" instruction. This would be done in the following fashion.

```
CALL MYSUB, IO 22=1
```

This would call up a subroutine labled "MYSUB" when I/O line 21 is active.

```
BR 200, IO 22=0
```

This would branch to address 200 when I/O line 22 is inactive.

Writing to an output is accomplished by entering the following into a terminal or program:

```
IO 21=1
IO 21=0
```

This would change the state of I/O line 21.

Read/Write an I/O Group

When using the IO variable to read the state of a group of inputs/outputs, or write to a group of outputs you would first want to configure the entire I/O group to be general purpose inputs or outputs using the IOS variable. In this case the response or input won't be a logic state of 1 or 0, but rather the decimal equivalent (0 to 63) of the 6 bit binary number represented by the entire group.

When addressing the I/O as a group the LSB (*Least Significant Bit*) will be line 1 of the group, (*e.g. 21, 31, 41, 51*). The MSB (*Most Significant Bit*) will be line 6 of the group (*e.g. 26, 36, 46, 56*).

The table on the left shows the bit weight of each I/O line in the group. It also illustrates the state should 6 LED's be connected to I/O group 20 when entering the IO variables in this exercise.

Configure the IOS variable such that group 20 is all general purpose outputs, active low or:

```
IOS 20 = 0,1,0
```

Enter the following in the terminal:

```
IO 20 = 35
```

As shown in the table I/O lines 26, 22 and 21 should be illuminated, 25, 24 and 23 should be off.

Enter this next:

```
IO 20 = 7
```

Now I/O 21, 22 and 23 should be illuminated.

```
IO 20 = 49
```

I/O 26, 25, and 21 are illuminated.

BIT WEIGHT DISTRIBUTION TABLE FOR GROUP 20 I/O					
I/O 26 MSB	I/O 25	I/O 24	I/O 23	I/O 22	I/O 21 LSB
32	16	8	4	2	1

BINARY STATE OF I/O GROUP 20 IO 20 = 35					
○	●	●	●	○	○
1	0	0	0	1	1
I/O 26 MSB	I/O 25	I/O 24	I/O 23	I/O 22	I/O 21 LSB

BINARY STATE OF I/O GROUP 20 IO 20 = 7					
●	●	●	○	○	○
0	0	0	1	1	1
I/O 26 MSB	I/O 25	I/O 24	I/O 23	I/O 22	I/O 21 LSB

BINARY STATE OF I/O GROUP 20 IO 20 = 49					
○	○	●	●	●	○
1	1	0	0	0	1
I/O 26 MSB	I/O 25	I/O 24	I/O 23	I/O 22	I/O 21 LSB

Table 8.3: Binary State of Outputs

NOTE: You can only write to General Purpose Outputs. If you attempt to write to and input or dedicated output type an error will occur!

SECTION 9

Configuring and Using the Expansion Modules

Section Overview

This section covers the configuration and usage of the optional expansion modules available for the MicroLYNX System. For instructions on installing the expansion modules into your MicroLYNX System please see *Section 2: Installing and Mounting the MicroLYNX*, of this document. The modules covered in this section are:

- The Isolated Digital I/O Module.
- The High-Speed Differential I/O Module.
- Typical Functions of the Differential I/O.
- The Analog Input/Joystick Interface Module.
- Typical Functions of the Analog Input/Joystick Module
- The RS-232 Expansion Module (CAN only).
- The RS-485 Expansion Module (CAN only).

MicroLYNX Expansion Modules

Additional Isolated Digital I/O

The Isolated Digital I/O can be expanded an additional 3 groups (30 - 50) for a total of 24 programmable I/O lines.
These expansion boards can go in any available slot. The group number will be determined by whichever slot they are plugged into: slot 1 will be group 30, slot 2 will be group 40, and slot 3 will be group 50.
These expansion boards will be configured and used in the same manner as the I/O bank that is standard on the MicroLYNX.
The IMS Part # for this item is MX-DI100-000 (Terminal Block) or MX-DI200-000 (10 Pin Header).

High-Speed Differential I/O Module

If closed loop motion control, ratio functions such as following or electronic gearing or the ability to sequentially control a second axis is required, up to two High-Speed Differential I/O Modules can be installed in slots 2 and 3, giving three channels of high-speed differential (or single) I/O apiece. The IMS Part # for this item is MX-DD100-000 (Terminal Block) or MX-DD200-000 (Pin Header).

Analog Input/Joystick Module

The Analog Input/Joystick Module features two 12 bit, 0 to 5 volt input channels which can be used to monitor devices such as temperature and pressure sensors. It can also be used to control an axis with a joystick. It features two voltage outputs: a 5 volt joystick reference, and a precision 4.096 volt calibration reference. This device can be installed in any available slot.
The IMS Part # for this item is MX-AJ100-000 (Terminal Block) or MX-AJ200-000 (Pin Header).

Choosing the Expansion Modules for Your Application

A powerful feature of the MicroLYNX is the versatility offered by its wide range of configurations available through the expansion modules.
The expansion modules listed above may be used singly or in combination to customize your MicroLYNX System to the specific requirements of your application. The table on the following page lists a collection of possible application requirements and their suggested MicroLYNX configurations.

MicroLYNX Expansion Slot Configuration				
Application Requirement	Isolated Digital I/O	High Speed Differential	Analog Input/Joystick	Expansion Slot
6 Additional Isolated I/O	X			SLOT 1
				SLOT 2
				SLOT 3
12 Additional Isolated I/O	X			SLOT 1
	X			SLOT 2
				SLOT 3
18 Additional Isolated I/O	X			SLOT 1
	X			SLOT 2
	X			SLOT 3
Encoder Feedback				SLOT 1
		X		SLOT 2
				SLOT 3
Encoder Feedback + Secondary Clock Out or In				SLOT 1
		X		SLOT 2
		X		SLOT 3
6 Additional Isolated I/O + Encoder Feedback	X			SLOT 1
		X		SLOT 2
				SLOT 3
12 Additional Isolated I/O + Secondary Clock Out or In	X			SLOT 1
		X		SLOT 2
	X			SLOT 3
Analog Input or Joystick Control				SLOT 1
				SLOT 2
			X	SLOT 3
6 Additional Isolated I/O + Analog Input or Joystick Control	X			SLOT 1
				SLOT 2
			X	SLOT 3
12 Additional Isolated I/O + Analog Input or Joystick Control	X			SLOT 1
	X			SLOT 2
			X	SLOT 3
Analog Input or Joystick Control + Encoder Feedback				SLOT 1
		X		SLOT 2
			X	SLOT 3
6 Additional Isolated I/O + Analog Input or Joystick Control + Encoder Feedback	X			SLOT 1
		X		SLOT 2
			X	SLOT 3

Table 9.1: MicroLYNX Expansion Module Configurations

Expanding the Isolated Digital I/O

The Isolated Digital I/O can be expanded to 24 lines. Expansion to this level would require the use of all three slots. The I/O groups are slot dependent. The slots will yield the following groups as numbered:

Slot 1 ..... Group 30
Slot 2 ..... Group 40
Slot 3 ..... Group 50

Pin #	Connector Option					
	8 Position Phoenix			10 Pin Header		
	Slot 1	Slot 2	Slot 3	Slot 1	Slot 2	Slot 3
1	V_{pullup}	V_{pullup}	V_{pullup}	IO 31	IO 41	IO 51
2	IO 31	IO 41	IO 51	IO 32	IO 42	IO 52
3	IO 32	IO 42	IO 52	V_{pullup}	V_{pullup}	V_{pullup}
4	IO 33	IO 43	IO 53	IO 33	IO 43	IO 53
5	IO 34	IO 44	IO 54	N.C.	N.C.	N.C.
6	IO 35	IO 45	IO 55	IO 34	IO 44	IO 54
7	IO 36	IO 46	IO 56	N.C.	N.C.	N.C.
8	I/O GND	I/O GND	I/O GND	IO 35	IO 45	IO 55
9				I/O GND	I/O GND	I/O GND
10				IO 36	IO 46	IO 56

Table 9.2: Isolated Digital I/O Group and Line Locations by Connector Option and Slot

Installing The Isolated Digital I/O Module

To install the Isolated Digital I/O Expansion Module in your MicroLYNX perform the following in accordance with Figure 9.1.

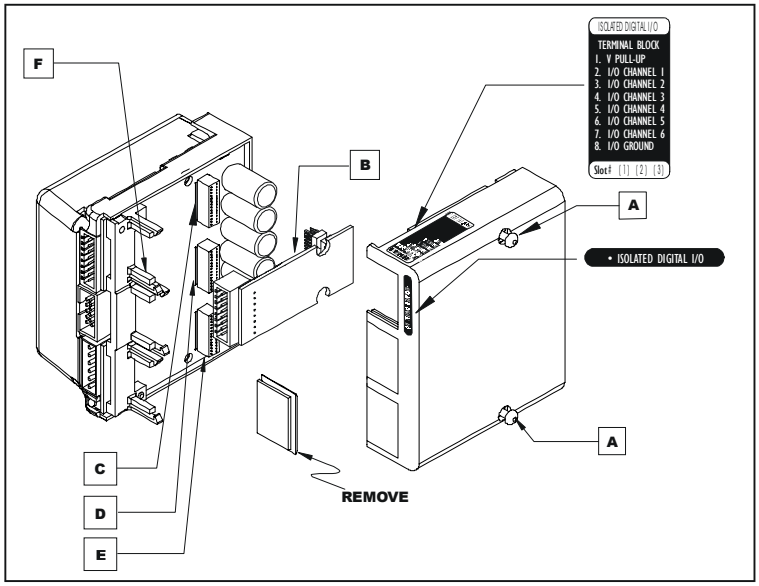


Figure 9.1: Installing the Isolated Digital I/O Expansion Module

1] Remove screws (A).
2] Remove panel from slot to be used.
3] Insert Isolated Digital I/O Module into Slot 1 (C), Slot 2 (D) or Slot 3 (E).
4] Press firmly until expansion board is securely seated and locked into place by retaining clips (F).
5] Reassemble MicroLYNX case in accordance with Figure 9.1.
6] Affix labels as shown. Use a highlighter or marker pen to highlight slot(s) used.

Using the Isolated Digital I/O

The Isolated Digital Expansion I/O operates in the very same manner as the standard isolated I/O. The only differences are the location of the pull-up switches, and the method of supplying an external pull-up voltage.

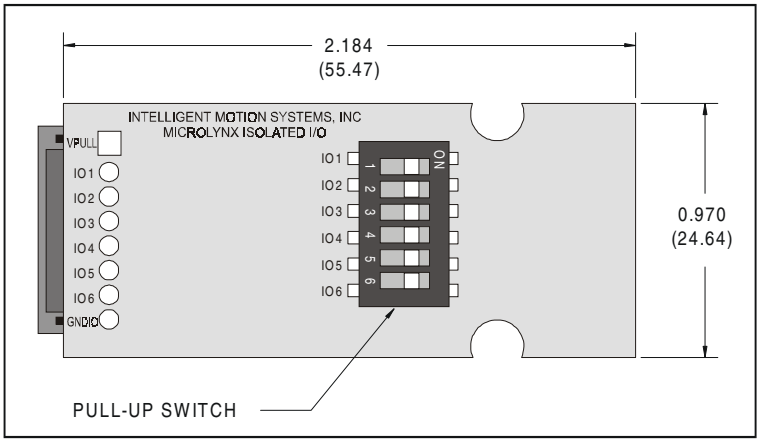


Figure 9.2: The Isolated Digital I/O Module, Bottom View

The pull-up switches are located on the bottom of the expansion board. They operate in the same fashion as the standard I/O set pull-ups. Configuring and using these switches is detailed in Section 8 of this document. Another key difference is the method by which an external pull-up voltage is supplied to the I/O. While the I/O Ground is common to each Isolated Digital I/O Module installed (both the Differential I/O Module and the Analog Input/

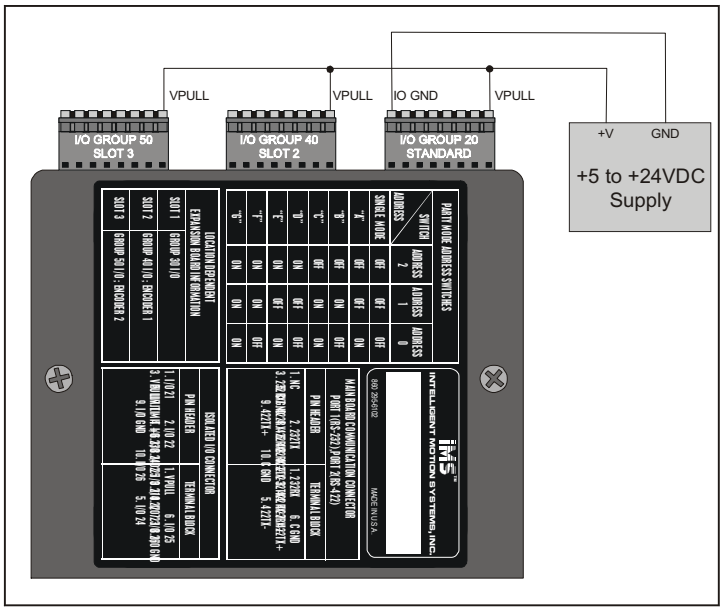


Figure 9.3: Powering Multiple Isolated Digital I/O Modules

Joystick Module have separate, non-isolated grounds) V-PULLUP is **NOT** common. This allows you to power each I/O group independently if you choose.

The expansion isolated digital I/O is configured and controlled by the IOS variable and the IO instructions in the same manner as the standard I/O set. The only difference is in how the lines and groups are addressed.

See Section 8 for instructions on using the isolated I/O

.

If digital filtering is used (IOF variable) it must be configured for each group separately.

The High-Speed Differential I/O Module

Figure 9.4: The Differential I/O Module

The MicroLYNX has the capability of having up to two High-Speed Differential I/O Modules installed in expansion slot numbers 2 and 3. The High-Speed Differential I/O Module expands the capabilities of the MicroLYNX to include application features such as:

1] Closed Loop Motion Control (Encoder Feedback).
2] Electronic Gearing (Ratio Functions).
3] Secondary Clock Output.
4] General Purpose High-Speed I/O.

High-Speed Differential I/O Expansion Board Electrical Characteristics	
Specification	
Max. Clock Frequency	5MHz
Digital Input Filtering	39kHz to 5MHz
Encoder Power	+5VDC Output
Current Limit (All Outputs Combined)	150mA
Slot Position	2 or 3
Max. # Usable Modules	2

Table 9.3: Electrical Characteristics

Pin #	Connector Option			
	8 Position Phoenix		10 Pin Header	
	Slot 2	Slot 3	Slot 2	Slot 3
1	I/O 17(-)	I/O 18(-)	N.C.	N.C.
2	GND	GND	+5 VDC	+5 VDC
3	+5 VDC	+5 VDC	GND	GND
4	I/O 14(-)	I/O 16(-)	I/O 14(-)	I/O 16(-)
5	I/O 13(+)	I/O 15(+)	I/O 13(-)	I/O 15(-)
6	I/O 14(+)	I/O 16(+)	I/O 13(+)	I/O 15(+)
7	I/O 17(+)	I/O 18(+)	I/O 14(-)	I/O 16(-)
8	I/O 13(-)	I/O 15(-)	I/O 14(+)	I/O 16(+)
9			I/O 17(-)	I/O 18(-)
10			I/O 17(+)	I/O 18(+)

Table 9.4: High-Speed Differential I/O Expansion Pinout by Connector Style and Slot

The pinout by slot location and connector style is given in Table 9.4.

The high-speed differential I/O is non-isolated, meaning the ground is not common with the isolated I/O ground.

Installing the High-Speed Differential I/O Module

To install the High-Speed Differential I/O Expansion Module in your MicroLYNX perform the following in accordance with Figure 9.5.

1] Remove screws (A).
2] Remove panel from slot to be used (either Slot #2 or Slot #3).
3] Insert High-Speed Differentiall I/O Module into Slot 2 (C), or Slot 3 (D).
4] Press firmly until expansion board is securely seated and locked into place by retaining clips (F).
5] Reassemble MicroLYNX case in accordance with Figure 9.1.
6] Affix labels as shown. Use a highlighter or marker pen to highlight slot(s) used.

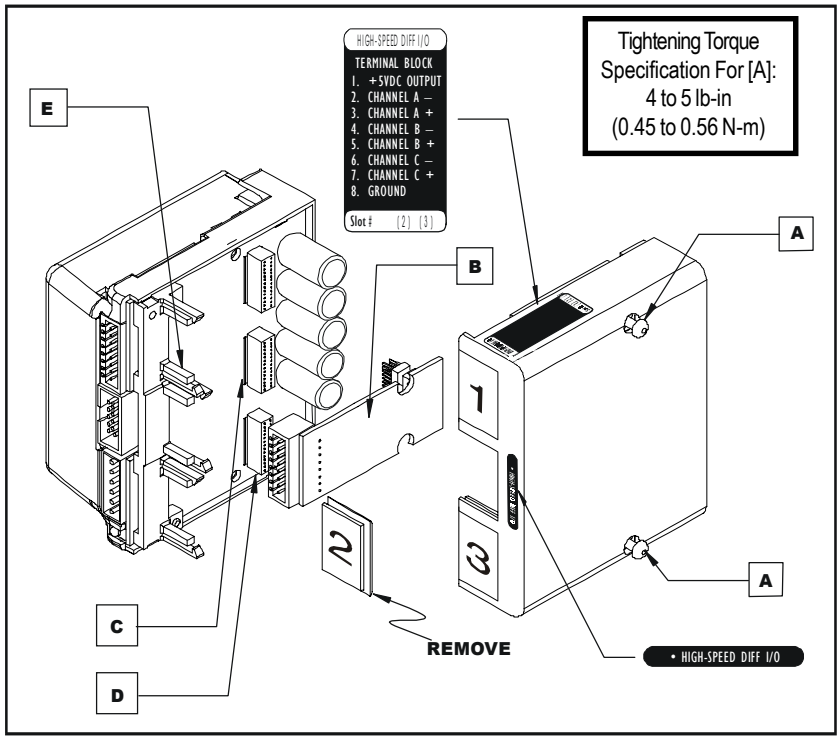


Figure 9.5: Installing the High-Speed Differential I/O Expansion Module

The Four Clocks Explained

The MicroLYNX has four clock pairs that are used by the high-speed I/O. One of these, clock pair 11 and 12, is fixed as an output and is used internally to provide step clock and direction pulses to the driver section of the MicroLYNX. The step clock output increments CTR1 (Counter 1). The user has no physical access to this clock, however, CTR1 may be read from or written to by software instructions in either program or immediate mode. The following table explains the clocks, as well as their default I/O line pair placement:

Clock Types Defined

There are three basic types of clocks that may be configured for the MicroLYNX, they are:
1] Quadrature

2] Step/Direction

3] Up/Down

These clock functions are illustrated in figure 9.6.

Quadrature

The quadrature clock function is the most commonly used input clock function. This is the default setting for each high-speed I/O channel except 11 & 12. This clock function will typically be used for closed loop control (encoder feedback) or for following applications

Step/Direction

The step/direction clock funtion would typically be used in an application where a secondary or tertiary clock output is required to sequentially control an additional axis.

Up/Down

The up/down clock type would typically be used as an output function where a secondary axis is being driven by a stepper or servo drive with dual-clock direction control circuitry.

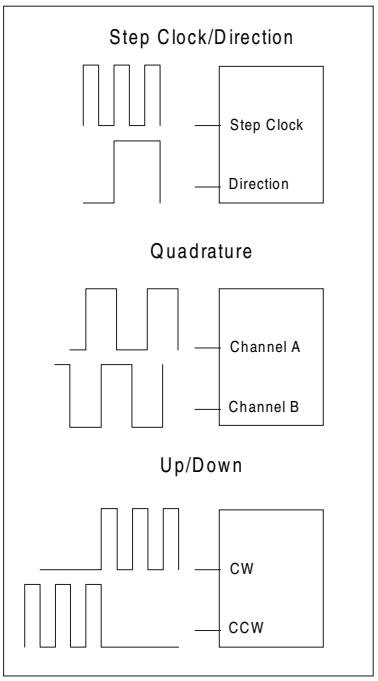


Figure 9.6: Clock Functions

The Four Clocks				
Clock #	**I/O Line Pair**	**Slot Position**	**Counter**	**Function**
1	11 & 12	None	CTR1	This clock is internally generated motion clock. It provides step clock and directional control to the driver section. This clock is not available on any external connector.
2	13 & 14	Slot 2	CTR2	May be configured as an input or output. By default this is configured as a quadrature input. It can be configured as a secondary clock output electronically geared to CLK1.
3	15 & 16	Slot 3	CTR3	May be configured as an input or output. By default this is configured as a quadrature input. It can be configured as a tertiary clock output electronically geared to CLK1.
4	17	Slot 2	None	May be configured as a high speed input or an output. As an output it is a 1MHz reference clock.
	18	Slot 3	None	May be configured as a high speed input or output. As an output it is a 10MHz reference clock.

Table 9.5: The Four Clocks and Their Default Line Placement

Configuring the Differential I/O - The IOS Variable

The high-speed differential I/O is configured by means of the IOS variable, and is used in the the same fashion in which the isolated I/O is configured. The main difference lies in that there are three additional parameters which need to be set in configuring the triggering, clock type and ratio mode setting.

It is important to note that the high-speed differential I/O lines may be used for the same input or output functions as the isolated digital I/O where the higher speed capabilities of the differential I/O is required.

However, for purposes of this example we will only illustrate the clock functions associated with the high-speed differential I/O. Figure 9.6 illustrates the IOS variable settings for the high speed differential I/O.

Configuring the High Speed I/O a Non-Clock Function

Configuring the high speed I/O to clock functions will be covered in depth in the following subsections on configuring encoder and ratio functions. Here we will briefly discuss using the high speed I/O as a general purpose or dedicated I/O function.

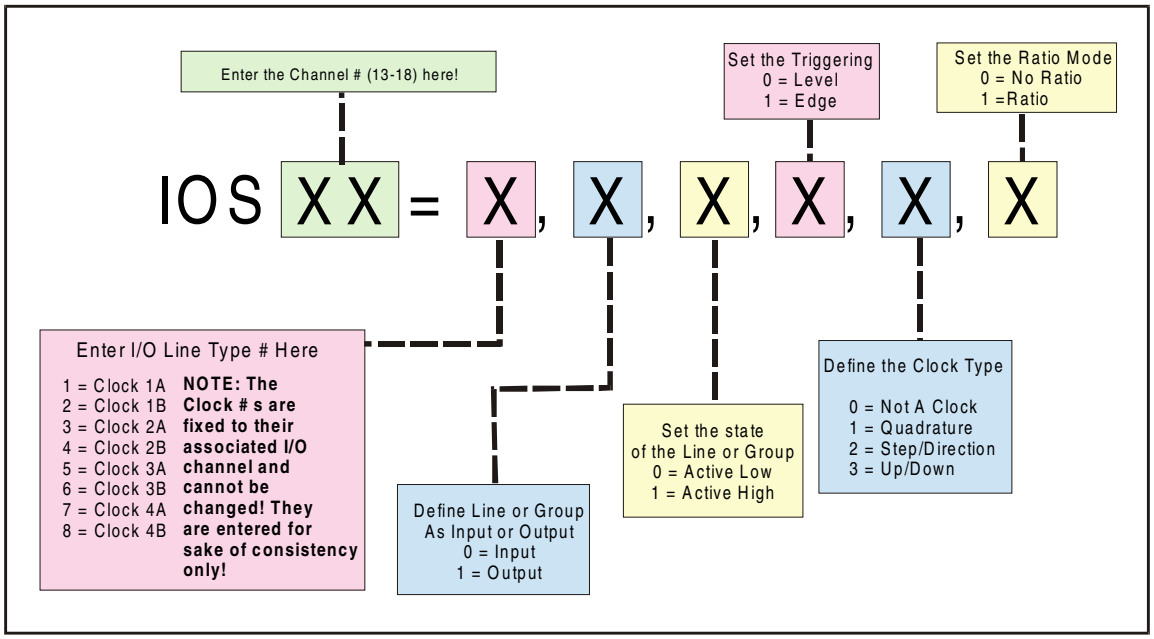


Table 9.6: IOS Variable Settings for the High-Speed Differential I/O

Care must be taken when configuring the high speed I/O to a general purpose or dedicated function as the output current sink is 150mA for the entire I/O group 10.
The IOS variable will be configured for the high speed I/O in the same fashion as it is set for the isolated I/O.

Configuring an Input

Clocks 2, 3 and 4 can be configured as high speed inputs, or as a general purpose input in the same fashion as the Isolated I/O. In configuring the Differential I/O line as a general purpose input you would typically use the "+" line of the line pair. You cannot use both lines as separate I/O lines. The figure below shows the

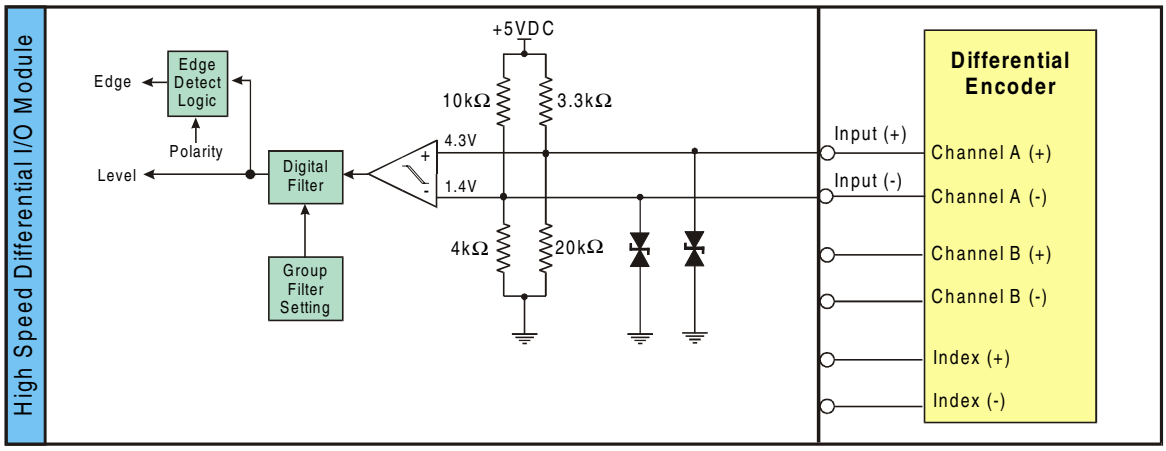


Figure 9.7: Differential I/O Input Equivalent Circuit

Input Equivalent Circuit with the I/O line pair connected to channel A of a differential encoder. This feature is demonstrated in *Typical Functions of the Differential I/O: Connecting and Using an Encoder.* Clocks 2, 3 and 4 are set up as Quadrature inputs by default. The defaults for each I/O Line Pair are:

IOS 13 = 3, 0, 1, 0, 1, 0
IOS 14 = 4, 0, 1, 0, 1, 0
IOS 15 = 5, 0, 1, 0, 1, 0
IOS 16 = 6, 0, 1, 0, 1, 0
IOS 17 = 7, 0, 1, 0, 1, 0
IOS 18 = 8, 0, 1, 0, 1, 0

Setting the Digital Input Filtering for the Differential I/O

User-definable digital filtering makes the LYNX well suited for noisy industrial environments. The filter setting is software selectable using the *IOF Variable* with a minimum guaranteed detectable pulse width of 18 microseconds to 2.3 milliseconds. Table 9.7 illustrates the IOF settings.

IOF Filter Settings for the High Speed Differential I/O IOF=<num> (<num> = 0-7)		
Filter Setting	**Cutoff Frequency**	**Minimum Detectable Pulse Width**
0 (default)	5.00 MHz	100 nanoseconds
1	2.50 MHz	200 nanoseconds
2	1.25 MHz	400 nanoseconds
3	625 kHz	800 nanoseconds
4	313 kHz	1.6 microseconds
5	156 kHz	3.2 microseconds
6	78.1 kHz	6.4 microseconds
7	39.1 kHz	12.8 microseconds

Table 9.7: Digital Filter Settings for the Differential I/O

Configuring an Output

The Differential I/O Group 10 has 3 Channels (Line Pairs 13 & 14, 15 & 16, and 17 & 18) that can be configured as an output by the user and One Channel (Line Pairs 11 & 12) that is configured as output only. (SCK and DIR on the Control Module.) These outputs can be configured as high speed outputs or 0 to 5VDC general purpose outputs by using the IOS variable. The high speed clock outputs have the following restrictions:

Line Pairs 11/12, 13/14 and 15/16 can be configured to Step Clock/Direction or Up/Down.

Line Pair 17/18 is limited to 1MHz Reference Out (17) and 10MHz Reference Out (18).

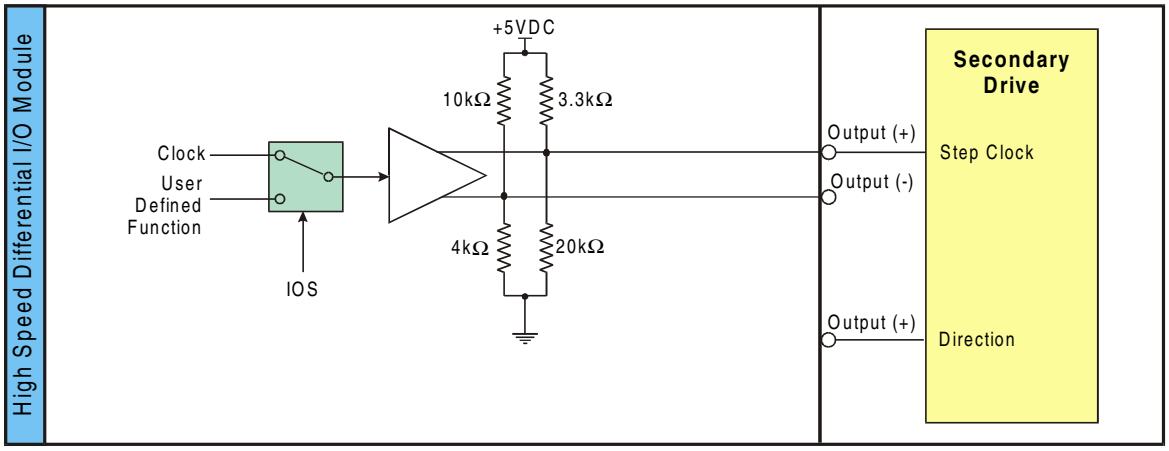


Figure 9.8: Differential I/O Output Equivalent Circuit

In the Equivalent Circuit in Figure 17 an Output is being used as Step or Direction on a driver.

For the configuration example, use I/O line 13 for the output. Since by default the line is a quadrature input we must configure it to be a Step/Direction Output by setting the IOS Variable to the following:
IOS 13 = 3, 1, 0, 1, 2, 0
This breaks down as:

IOS 13 - Identifies the line being configured as 13.
3 - Sets the I/O Type to Clock 2A (default).
1 - Sets it as an output.
0 - Sets Logic at Low True.
1 - Edge Triggered.
2 - Sets the Clock Type to Step/Direction.
0 - No Ratio.

Typical Functions of the Differential I/O

Connecting and Using an Encoder

The high-speed differential I/O module may be used for closed loop motion control by receiving quadrature input from a differential or single ended encoder.
High-Speed I/O channels 13 and 14 are configured by default for this function, so you would want your expansion module inserted into expansion slot #2.
Connect your encoder as shown in the following figure and table.

Encoder Connections - Expansion Slot #2				
Encoder Signal		MicroLYNX		
Differential	Single	I/O Channel	8 Position Pheonix	10 Pin Header*
Channel A+	Channel A	13+	5	6
Channel A-		13-	8	5
Channel B+	Channel B	14+	6	8
Channel B-		14-	4	4 or 7
Index +	Index	17+	7	10
Index -		17-	1	9
+5 VDC	+5 VDC		3	2
GND	GND		2	3
NOTE: IMS differential encoder follow the Hewlett Packard pin configuration. Thus if your encoder is manufactured by HP or IMS a 10 conductor "straight through wired" ribbon cable with female DIN ribbon cable connectors can be connected directly between the differential encoder and the expansion board (10 Pin Header Version) without wiring modification.				

Table 9.8: Expansion Slot 2 Encoder Connections

Testing Your Encoder Setup

Now that your encoder is connected, let's test the setup and verify its operation by typing the following into your terminal:

```
'set munits to correspond with MSEL=256
MUNIT=51200

'set the encoder units variable EUNIT to the number = 4 x
'encoder resolution, ie 500 line encoder x 4 = 2000,
```

```
'200 line encoder x 4 = 800 etc.
EUNIT=2000

'Set the stall factor variable to 10% of EUNIT (10% of a
'revolution
STLF=200

'Enable encoder functions
EE=1
POS=0  'set position counter to 0
CTR2=0 'set counter 2 to 0
SAVE   'save the aforementioned settings.
```

Test the encoder setup by entering the following into your terminal:

```
MOVR 10        'the motor moves 10 revolutions (we hope)
PRINT POS      'we read the POS variable, it should say "10.000"
PRINT CTR2     'we read CTR2, it should read 10 X EUNIT, or 20000
```

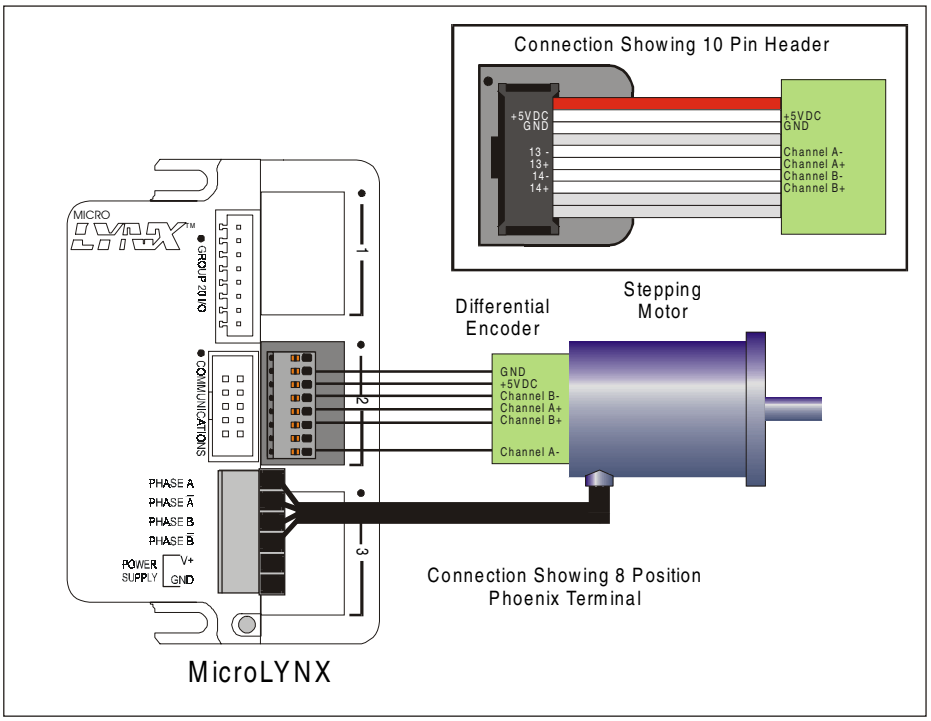


Figure 9.9: Differential Encoder Connection

Introducing The EUNIT (Encoder UNITS) Variable

During open loop operation, the MicroLYNX takes the number of clock pulses registered on CTR1, scales that number using the MUNIT variable and then writes the result to the position variable POS.
For closed loop operation, where the encoder functions are enabled (EE=1), the MicroLYNX takes the number of clock pulses registered on CTR2, scales them by the EUNIT variable and stores them to the POS counter.

The EUNIT variable must be scaled to the same factor as the MUNIT variable. For example, if you were scaling your system to operate in degrees, the MUNIT/EUNIT relationship would be expressed thus:

```
MUNIT=51200/360
```

```
EUNIT=2000/360
```
(This assumes MSEL=256 and a 500 line encoder.)
With this configuration if you performed the following absolute move:
```
MOVA 270
```
the axis would turn 270°. Thus when you enter:
```
PRINT POS
```
the terminal will display "270.00".
The program that follows will illustrate encoder feedback by making a series of moves while displaying both the raw counts from CTR2 and the scaled POS value.
Enter the program below in the text editor window.

```
'******PARAMETERS*******
MUNIT=51200'motor units = 1/256 resolution
EUNIT=2000 '500 line encoder quad input
EE=1       'enable encoder functions
STLF=200          'stall factor 10% of 1 rev.
STLDE=1           'enable stall detection
STLDM=0           'stop motion if stall is detected
MAC=75            'accel. current to 75%
MRC=50            'run current to 50%
MHC=25            'hold current to 25%


'******PROGRAM********
PGM 200
  CTR2=0
  POS=0
  MOVR 1
  HOLD 2
  DELAY 250
  PRINT "\rEncoder Count= ", CTR2, "  Position Count= ", POS,"\e[K";
  MOVR 10
  HOLD 2
  DELAY 250
  PRINT "\rEncoder Count= ", CTR2, "  Position Count= ", POS,"\e[K";
  MOVR -11
  HOLD 2
  DELAY 250
  PRINT "\rEncoder Count= ", CTR2, "  Position Count= ", POS,"\e[K";
  BR 200
END
PGM
```

Execute the program by entering "EXEC 200" into the terminal.

Following an External Clock (Electronic Gearing)

The High-Speed Differential I/O Module allows you to configure the MicroLYNX's primary axis to follow an external clock input. The hardware connection *(Figure 9.8)* is almost identical to that shown for closed loop control, only in this instance instead of using a quadrature clock input for position monitoring and maintenance, we will use the encoder input to control the primary axis.

Using this type of application introduces the HAE (Half Axis Enable) flag and the HAS (Half Axis Scaling) variable. In half axis mode the master clock is taken from the CLK2, CLK3 or CLK4 (I/O channels 13 & 14, 15 & 16 or 17 & 18), which have the IOS variable configured as inputs, a clock type, and ratio mode enabled. The primary axis will move as a ratio of this clock based upon the factor entered in the HAS variable.

HAE Half Axis Enable/Disable Flag

This flag (1) enables and (0) disables half axis scaling mode. The default condition is (0) disabled. The HAE flag must be enabled for this mode to function.

HAS Half Axis Scaling Variable

The half axis scaling variable is the factor by which the Follower Input: Primary Axis ratio is scaled. The range of the factor is >-1 to <1. For example, a setting of HAS=.5 will output 1 pulse on the primary axis for every 2 pulses input to the follower input or a 2:1 ratio, HAS=.2 will be 5:1, HAS=.999 will be .999:1 and so on. The default HAS value is 0.000, thus some factor must be entered to make this function.

Configuring the I/O for Half Axis Mode

The parameter setup to make this configuration follows. This assumes a High-Speed Differential I/O Expansion Module installed in slot 2. If your module is installed in slot 3, use I/O channels 15 and 16 (IOS 15=5,0,1,0,1,1 and IOS 16=6,0,1,0,1,1) instead. The raw count of clock pulses will register to CTR3. I/O channels 17 and 18 can be used for this also, only there is no registration of clock pulses:

```
IOS 13=3,0,1,0,1,1    'I/O 13 quad. input, ratio mode
IOS 14=4,0,1,0,1,1    'I/O 14 quad. input, ratio mode
HAE=1                 'Enable half-axis scaling mode
HAS=.5                'Half-axis scaling variable to .5 (1 output
                      'pulse on the pri. axis for 2 input pulses)
```

With this configuration, one (1) step clock pulse will output to the primary axis for every two (2) input clock pulses.
By reading the value of CTR2 and CTR1 you can see the ratio of the pulses.
Try different HAS variable, motor resolution and MUNIT settings to see how the primary axis is effected by different settings.

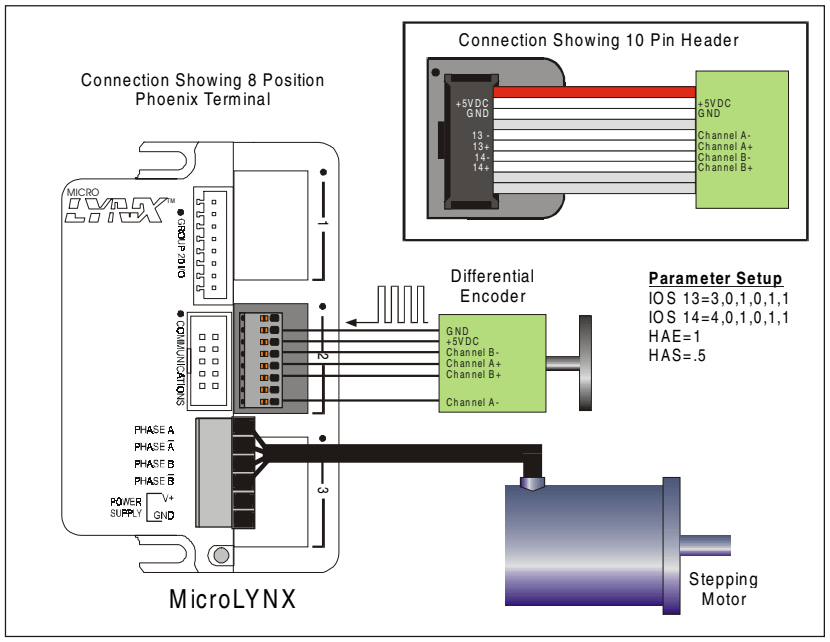


Figure 9.10: Differential I/O Connections for Following an External Input

NOTE: The HAS variable must be set to less than 1 or Error Code 9004, “Ratio Out of Range” will occur,

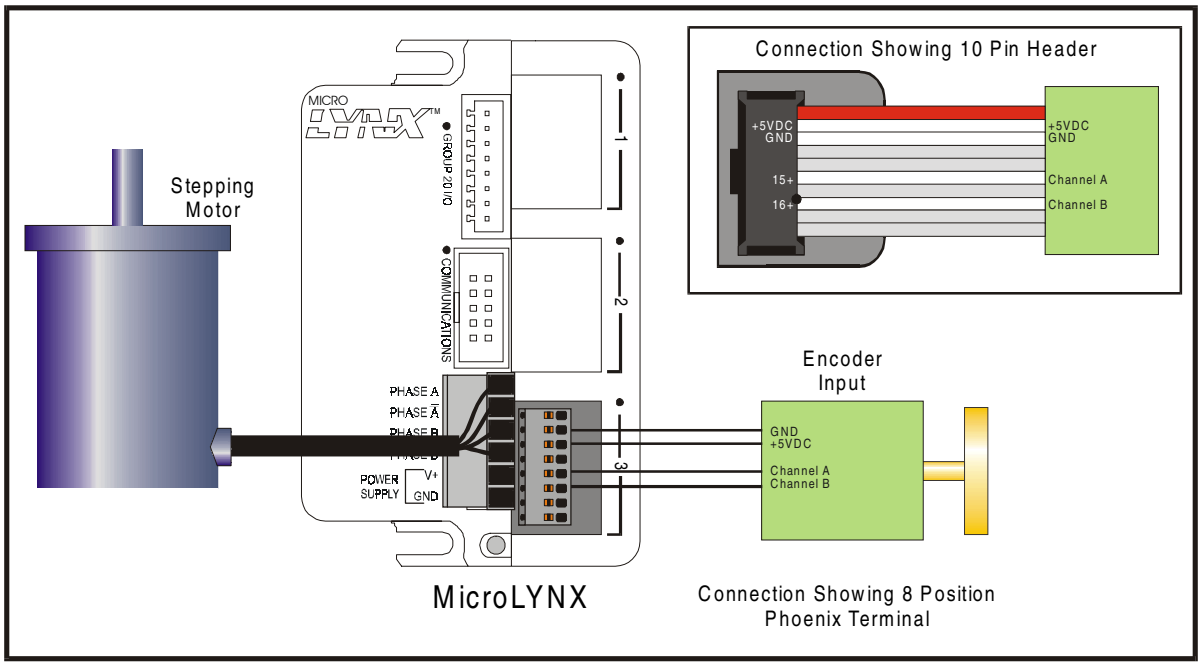


Figure 9.11: One and a Half Axis Operation

One and a Half Axis Operation (RATIOE)

A secondary drive can be connected to a pair of differential outputs. The secondary driver will operate off of the differential output pair 15 and 16 (I/O pair 13 and 14 can also operate in this mode). Setting the ratio mode to TRUE (1) for the differential output clock (IOS) specifies a secondary drive function. Then when ratio mode is enabled (*RATIOE*); the secondary axis will follow the primary axis with the ratio specified by the *RATIO* variable.

The sequence of commands used to make this setup function would be as follows:

```
'Set IOS 15 to step/direction clock type, and ratio mode
IOS 15 = 5,0,1,0,2,1
'Set IOS 16 to step/direction clock type, and ratio mode
IOS 16 = 6,0,1,0,2,1
'Set Ratio Mode Enable Flag to TRUE (1)
RATIOE = 1
'Set RATIO variable to .5 for the secondary drive
RATIO = .5
```

With this setup, the motor on the secondary drive will move half the distance of the primary.

NOTE: The RATIO variable must be set to less than 2 or -2 or Error Code 9004, "Ratio Out of Range" will occur,

The Analog Input/Joystick Module

The Analog Input/Joystick Module adds two 0 to 5 volt analog input channels to the MicroLYNX System. Both channels can be used for data aquisition, or either channel can be used to directly control motion. This offers the user the capability of receiving input from a variety of analog sources such as temperature or pressure sensors, and then controlling events based upon those inputs.

The user-selected Joystick channel can be programmed to set the range, zero, deadband and sensitivity.

Each channel uses a 12 bit D/A converter for better resolution as well as a fixed single pole analog filter with a cutoff frequency of 658 Hz to reduce the electrical noise that can be present in industrial environments.

The Analog Input/Joystick Module can be installed in any free slot, however only one (1) module can be used per MicroLYNX.

Analog Input/Joystick Module Specifications	
Specification	
Analog Input Voltage Range	0 to +5 volts
Resolution	12 Bits
Offset	±3 LSB
Integral Linearity Error	±2 LSB
Differential Linearity Error	±3/4 LSB
Absolute Maximum Voltage at Inputs	±24 volts
Joystick Reference Voltage	+5 volts
Precision Calibration Reference Voltage	+4.096 volts ±0.2%
Calibration Reference Voltage Tolerance	±2%
Analog Input Filter Cutoff Freq.	658 Hz
Slot Position	1, 2 or 3
Maximum # Usable	1

Table 9.9: Analog Input Module Specifications

Pin #	Connector Option	
	8 Position Phoenix	10 Pin Header
1	+5V (Joystick Reference)	+5V (Joystick Reference)
2	AIN 1	GND
3	GND	AIN 1
4	+5V (Joystick Reference)	+5V (Joystick Reference)
5	AIN 2	GND
6	GND	AIN 2
7	4.096V (Calib. Reference)	4.096V (Calib. Reference)
8	GND	GND
9		GND
10		N.C.

Table 9.10: Analog Input/Joystick Module Pin Configuration

Installing the Analog Input/Joystick Module

To install the Analog Input/Joystick Expansion Module in your MicroLYNX, perform the following in accordance with Figure 9.9 .

1] Remove Screws (A).

2] Remove panel from slot to be used.

3] Insert Analog Input/Joystick Module into Slot 1 (C), Slot 2 (D) or Slot 3 (E).

4] Press firmly until expansion board is securely seated and locked into place by retaining clips (F).

5] Reassemble MicroLYNX case in accordance with Figure 9.9.

6] Affix labels as shown. Use a highlighter or marker pen to highlight slot used.

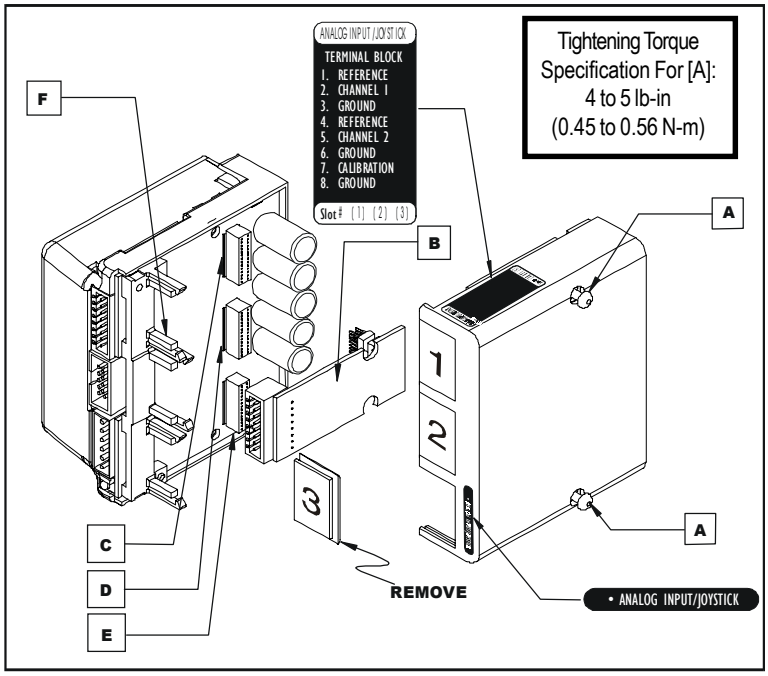


Figure 9.12: Installing the Analog Input/Joystick Module

Instructions & Variables Specific to the Analog Module

There are several new enhancements to the LYNX instruction set which add the functions of the Analog Input/Joystick Interface Module while maintaining backward compatibility with the modular LYNX System. The following instructions and variables are specific to the Analog Input/Joystick Interface Module (LYNX)

Analog Input/Joystick Interface Command Summary		
Instruction	**Usage**	**Description**
IJSC	**IJSC**	Supports the Analog/Joystick Interface Module when operating in joystick mode. Execution of this command followed by moving the connected joystick over its range of motion and back to center, and then pressing the "ENTER" key, or letting it time out for 30 seconds calibrates the joystick. This instruction allows for rapid calibration of the joystick.
Variable	**Usage**	**Description**
ADS	**ADS<chan>=<aunit>,<func>,<law>**	Setup variable supports the Analog Input/Joystick Module. **<chan>** = channel# (1 or 2) **<aunit>** = converts analog units to motor steps (velocity = aunit x munit) **<mode>** = 1 - analog, 2 - joystick **<law>** = Adjusts joystick position to motor velocity transformation. 1 = linear, 2 = square law, 3 = cube law
AIN	**<var>=AIN<chan>**	Causes a read of analog input channel **<chan>**, data is saved as variable **<var>**.
JSC	**JSC=<num>**	Joystick center position, updated by IJSC command or directly as shown.
JSDB	**JSDB=<num>**	Joystick deadband, updated by ISJC command or directly as shown.
JSFS	**JSFS=<num>**	Joystick Full Scale, updated by ISJC command or directly as shown.
Flag	**Usage**	**Description**
JSE	**JSE=<flg>**	Joystick Enable flag. <flg> = 1 enables joustick function, <flg> = 0 (default) disables.

Table 9.11: Analog Input/Joystick Software Command Summary

and Analog Expansion Module (MicroLYNX). These are introduced here and covered in more detail in the Software Reference.

Error Codes

In addition to the instructions and variables, the following error codes have been added to support the inclusion of the Analog Module and aid in troubleshooting the MicroLYNX System:

1201 Selected Analog Board not installed.
1202 Analog channel number not available.
1204 Analog option not installed.
1205 Analog VALUE out of range, possibly defective Board.

2101 Analog RANGE not allowed.
2102 Analog destination/source not allowed.
2103 Analog Destination/Source already used.
2104 Invalid Analog Channel number.
2105 Analog LAW not allowed.
2106 Can't enable Joystick while in motion, or can't exec motion cmd with Joystick enabled.

9014 Analog input not allowed for data.

The ADS Variable (A to D Setup)

The ADS variable is the heart of the MicroLYNX Analog Input/Joystick Interface Module. There are three parameters that control how the module will respond to input. It is used as follows:

```
ADS <chan>=<aunit>,<mode>,<law>
```

<chan>: Is the analog input channel that will be used, either 1 or 2.
<aunit>: This parameter sets the relationship between the analog input and units that are convenient to the user. In analog (User) mode the aunits parameter is the number of user units corresponding to the Analog Module full scale. In Joystick (Velocity) mode the aunits parameter is the number of munits/second corresponding to the Joystick Full Scale (JSFS) parameter.

<mode>: The mode parameter controls whether or not the input is used for velocity control; 1 = analog input, 2 = velocity or joystick mode.

<law>: Controls the sensitivity of the velocity with respect to the analog input. The effect of the analog input can be linear, square or cube. <law> applies to velocity mode only.
Here are two examples that illustrate the ADS variable:

Example 1

A pressure transducer is connected to input 1. The transducer output is 10 psi/volt. Vref represents the voltage at the Input to the Analog Joystick Module corresponding to full scale. Vref as measured at pin 1 on the Analog Joystick Module is 5.05 volts. Thus aunits for channel 1 is 10 psi/volt x 5.05 volts or 50.5. The value returned by an analog read of Channel 1 will be in psi. Note that the full scale output of the transducer does not have to equal the Analog Module full scale. This setup would be expressed thus:
ADS 1=50.5, 1

Example 2

A 1.8 degree (per full step) motor connected to a lead screw with a lead of .1 inches/rev. The step motor drive is set for 32 usteps per full step. A joystick is connected to channel 1. To program speed and motion in inches set munits to (32 pulses/1.8 degrees) x (360 degrees/1 rev) x (1 rev/.1 inches). If a maximum speed of 3 inches/second is desired while in Joystick operation set aunits for channel 1 to 3. For linear Joystick operation the setup command is ADS 1 = 3,2,1

Typical Functions of the Analog Input Module

There are three program examples that will illustrate the use of the Analog Input/Joystick Module. In each case a 1kΩ potentiometer is used to emulate a sensor for analog input mode, and a joystick for velocity mode.

Use the connection configuration shown in figure 9.13 below, a joystick or a sensor would be connected the same way.

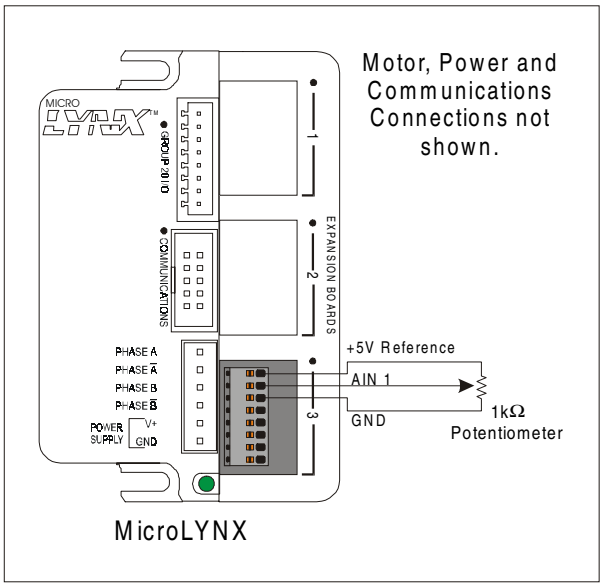


Figure 9.13: Analog Input Module Exercise Connection

Exercise 1: Velocity (Joystick) Mode

Here the potentiometer is emulating a joystick. Enter and execute the following program. When the voltage on AIN 1 is roughly 100mV either side of 2.5 volts it will be in the deadband range of the joystick. When less than 2.4 volts, the axis will accelerate in the minus (-) direction. When more than 2.6 volts, it will accelerate in the positive (+) direction. The velocity will increase as the voltage decreases from 2.4 to 0, or increases from 2.6 to 5.0. This can be watched with a multimeter. In this exercise both the axis velocity and position will display to the terminal screen.

```
'****Parameters****
MSEL=256
MRC=100
MAC=100
MUNIT=51200
JSDB=100          'Joystick deadband =100 aunits
VM =10000         'max velocity 10,000 munits/sec
ADS 1=1000,2,1    'chan. 1,aunits=1000, joystick, linear law
JSE = 1           'enable joystick functions

'****Program****

PGM 1
PRINT "\e[2J"
  LBL RUN
  PRINT "\e[1;1HInput Channel =    " , AIN
  PRINT "Axis Velocity =  " , VEL
```

```
  PRINT "Axis Position = " , POS
  BR RUN
END
PGM
```

Exercise 2: Sensor Input I

Here we pretend the potentiometer is a pressure transducer and use it to display a pressure value to the screen.

```
ADS 1=50.5,1            'set ADS to aunit=50.5,analog input mode

PGM 200
LBL PRNTPSI             'name program "PRNTPSI"
PRINT "\e[2J"           'ansi esc. sequence to clear display
PRINT "Pressure = ", AIN 1 , " PSI"
BR PRNTPSI       'loop to program beginning
END
PGM
```

Exercise 3: Sensor Input II

Once again our potentiometer is pretending to be a sensor. In this exercise the program will call up a subroutine based upon the voltage seen on AIN 1 and position the axis at an absolute position. The best analog to this example might be a flow control application.

```
'****Parameters****
MUNIT=51200             'munits=51200
MAC=75                  'acceleration current to 75%
MRC=50                  'run current to 50%
ADS 1=5,1               'aunits 5, analog input mode
VAR LIMIT=0             'declare user var "LIMIT"

'****Program****
PGM 200
LBL AINTST              'name program "AINTST"
LIMIT = AIN 1           'set user var "LIMIT" = AIN 1
CALL ATEST, LIMIT>3.5   'call ATEST if LIMIT is greater than 3.5 aunits
CALL BTEST, LIMIT<3.5   'call BTEST if LIMIT is less than 3.5 aunits
BR 200                  'loop to beginning of program
END

'****Subroutines****
LBL ATEST               'declare subroutine "ATEST"
VM=20                   'max. velocity = 20 munits/sec.
MOVA 10                 'index to abs. pos. 10
HOLD 2                  'suspend prog. until motion completes
RET                     'return from subroutine

LBL BTEST               'declare subroutine "BTEST"
VM=5                    'max velocity = 5 munits/sec.
MOVA 22                 'index to abs. pos. 22
HOLD 2                  'suspend prog. until motion completes
RET                     'return from subroutine
```

The RS-232 Port 2 Communication Expansion Module

The RS-232 Port 2 communications module, which allows for use of the RS-232 interface , can only be used with the CAN bus version of the MicroLYNX. This expansion board can be used in any of the three expansion slots and is automatically recognized by the MicroLYNX; no configuration is needed.

NOTE! Since the RS-232 Expansion Module uses MicroLYNX COMM 2, it cannot be used in conjunction with the RS-485 Expansion Module. Only one of the two interfaces can be used with the CAN bus version of the MicroLYNX.

This expansion board uses MicroLYNX COMM 2 and can be used to simultaneosly communicate with the MicroLYNX via RS-232, while communicating via the CAN bus. This is useful in requesting and displaying system status information from and to a PC or terminal.

RS-232 Expansion Module		
Pin #	Connector Option	
	8 Position Phoenix	10 Pin Header
1	CGND	N.C.
2	RS-232 RX	RS-232 TX
3	RS-232 TX	RS-232 RX
4	N.C	N.C.
5	N.C.	CGND
6	N.C.	N.C.
7	N.C.	N.C.
8	N.C.	N.C.
9		N.C.
10		N.C.

Table 9.12: RS-232 Expansion Pinout

The following table and diagram illustrate the pin configuration and connection of the RS-232 Expansion Module. See *Section 3: Installing and Mounting the MicroLYNX*, for installation instructions.

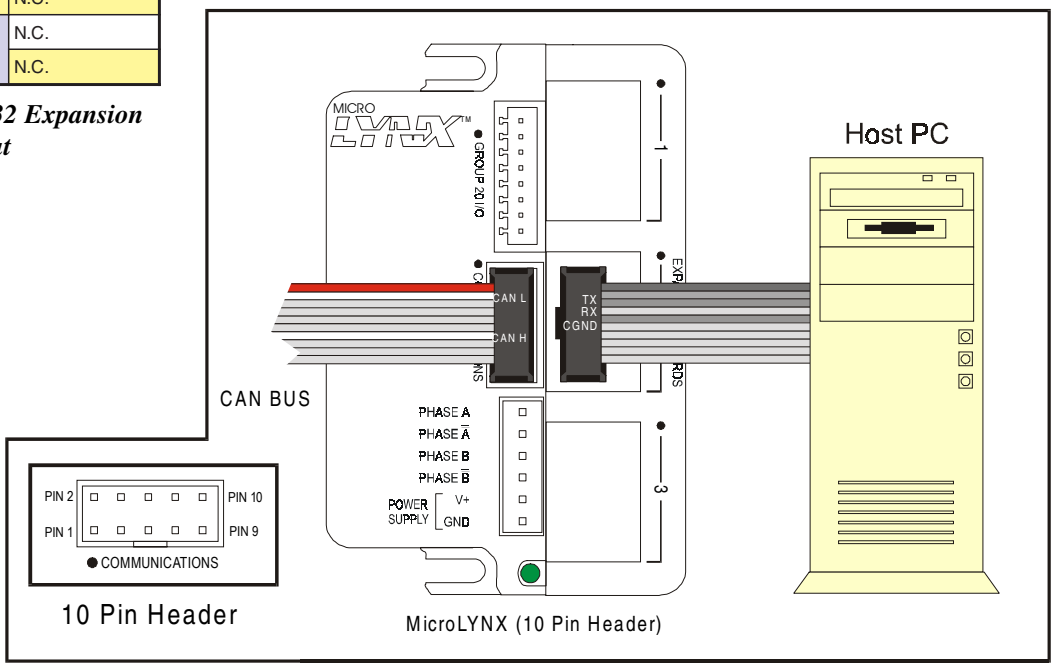


Figure 9.14: Connecting the RS-232 Expansion Module

The RS-485 Port 2 Communication Expansion Module

The RS-485 Port 2 communications expansion module allows for use of the RS-485 interface on the CAN bus version of the MicroLYNX only. This expansion board can be used in any of the three expansion slots and is automatically recognized by the MicroLYNX; no configuration is needed.

This expansion board uses MicroLYNX COMM 2 and can be used to simultaneosly communicate with the MicroLYNX via RS-485, while communicating via the CAN bus. This is useful in requesting and displaying

NOTE! Since the RS-485 Expansion Module uses MicroLYNX COMM 2, it cannot be used in conjunction with the RS-232 Expansion Module. Only one of the two interfaces can be used with the CAN bus version of the MicroLYNX.

system status information from and to a PC, terminal or machine interface such as the IMS HMI. It also allows for the use of multiple MicroLYNX Systems in a PARTY configuration without using additional CAN bus node positions, if the CAN interface is not required an all the MicroLYNX nodes. The following table and diagram illustrate the pin configuration and connection of the RS-485 Expansion Module. See *Section 3: Installing and Mounting the MicroLYNX*, for installation instructions. For multi-drop connection information see *Section 7: The Communications Interface*.

RS-485 Expansion Module		
Pin #	Connector Option	
	8 Position Phoenix	10 Pin Header
1	N.C.	N.C.
2	N.C.	N.C.
3	N.C.	N.C.
4	RS-485 RX-	N.C.
5	RS-485 RX+	CGND
6	RS-485 TX-	RS-485 RX+
7	CGND	RS-485 RX-
8	RS-485 TX+	RS-485 TX-
9		RS-485 TX+
10		CGND

Table 9.13: RS-485 Expansion Pinout

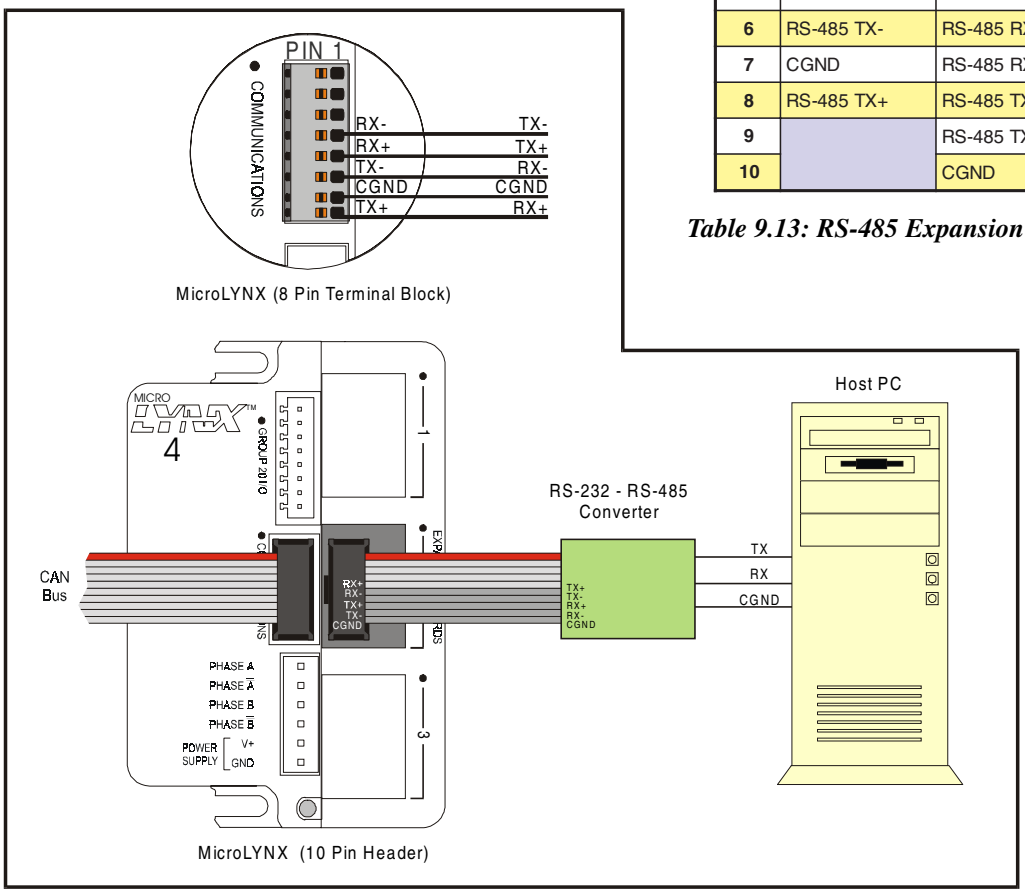


Figure 9.15: Connecting the RS-485 Expansion Module

This Page Intentionally Left Blank

Software Reference

Part III
Software Reference

Summary of Changes

The LYNX Terminal Software

Introduction to LYNX Programming

Functional Groups

Language Reference

ASCII Table

Error Table

Factory Defaults

Establishing Communication Using Hyperterminal

Table Of Contents

Summary of Changes 3-5

Software Enhancements 3-5
- New or Modified Instructions 3-5
- New Variables 3-6
- New Flags 3-7
- New Math/Logic Functions 3-7

Section 1: The LYNX Terminal Software 3-8

Section Overview 3-8
Installation and Setup 3-8
- System Requirements 3-8
- Installation 3-8

Using the LYNX Terminal Software 3-9
- Downloading A Program to the LYNX 3-10
- Uploading a Program form the LYNX 3-11
- Setting the Programmable Function Keys 3-11

Upgrading the Firmware in Your LYNX Product 3-12

Section 2: Introduction to LYNX Programming 3-13

Section Overview 3-13
Tools Required: 3-13
- Terminal 3-13
- Text Editor 3-14

Basic Components of Lynx Software 3-14
- Instructions 3-14
- Variables 3-14
- Flags 3-15
- Keywords 3-16

Most Commonly Used Variables and Commands 3-16
- Variables: 3-16
- Motion Commands 3-17
- I/O Commands 3-18
- System Instructions 3-19
- Program Instructions 3-19

Programming 3-21
- Program Samples 3-21

Section 3: Functional Grouping of the Instruction Set 3-25

Section Overview 3-25
- Using the Tables 3-25

Acceleration and Deceleration 3-26
Velocity 3-27
Position 3-27
Drive and Motor 3-28
Encoder 3-29
I/O 3-29
Miscellaneous Motion 3-30
Data 3-31
Event (Trip) 3-33
Instructions Which Can Be Used In A LYNX Program 3-34

Instructions Which Can Be Used In Immediate Mode 3-37
Miscellaneous And Setup Variables 3-39
Miscellaneous And Setup Flags 3-40
Mathematical And Logical Functions 3-41

Section 4: LYNX Programming Language Reference 3-42

Appendix A: ASCII TABLE 3-102

Appendix B: Error Table 3-104

Hardware Errors 3-104
I/O Errors 3-104
Clock Errors 3-105
Syntax Errors 3-105
Variable/Flag Errors 3-106
Motion Errors 3-106
Encoder Errors 3-106
NVM Errors 3-106
Out Of Range Errors 3-106

Appendix C: Factory Defaults 3-1108

Appendix D: Establishing Communications Using Windows95 Hyper Terminal 3-109

SUMMARY OF CHANGES

This part has undergone the following changes since the last revision:

1) The LYNX Terminal software section has been updated and moved from its prior location in Part I of this document to this part.
2) The Language Reference section has been reformatted.
3) The Error Table (Appendix A) has been updated to include support for the MicroLYNX.
4) The new instructions, variables, flags and math functions (listed below) have been added.

Software Enhancements

The following software enhancements have been incorporated into the LYNX System to provide increased functionality as well as to provide access to the new hardware features. These enhancements have been accomplished while maintaining backward compatibility with the Modular LYNX System. Enhancements are in the form of new instructions, system variables, flags and supporting error numbers, as well as expansion of existing instructions to provide new capabilities.

New or Modified Instructions

DVF

This is an enhancement of the DVF command in that it will now allow the deleting of variables and flags that are defined by the IP command. These new flags are:

HIGH,	H	=	1
LOW,	L	=	0
TRUE,	T	=	1
FALSE,	F	=	0
YES,	Y	=	1
NO,	N	=	0
OFF		=	1
ON		=	0

INPUT

This command is an enhancement of the existing command. The enhancement relates to the optional nowait parameter. If nowait is set to 1, the INPUT command will no longer suspend program execution waiting for an input. The Lynx program will continue executing and the var/flg will be updated whenever it is entered.

INPUT1

This is an enhancement of the INPUT command in that it will only accept input from COMM 1. Otherwise it operates the same way.

INPUT2

This is an enhancement of the INPUT command in that it will only accept input from COMM 2. Otherwise it operates the same way.

PRINT1

This is an enhancement of the PRINT command in that it will only output the print string to COMM 1. Otherwise it operates the same way.

PRINT2

This is an enhancement of the PRINT command in that it will only output the print string to COMM 2. Otherwise it operates the same way.

IJSC
This is a new command to support the Analog/Joystick module when operating in joystick mode. Execution of this command followed by moving the connected joystick over its range of motion and back to center, then pressing “Enter” or letting it time out in 30 seconds. This allows for rapid calibration of the joystick.

The TRIPs Have Been Renamed:

- ITx is now TIx - Trip on Input.
- ITEx is now TIEx - Trip on Input Enable.
- TPx is now TPx - Trip on Position.
- TPEx is now TPEx - Trip on Position Enable.
- TIx is now TTx - Trip on Timer.
- TIEx is now TTEx - Trip on Timer Enable.
- TIRx is now TTRx - Trip on Timer Repeat.
- VT is now TV - Trip on Velocity.
- VT is now TVE - Trip on Velocity Enable.

New Variables

ADS
Analog I/O setup supports the Analog/Joystick module.

DRVTP
Provides means to interrogate MicroLYNX to determine system configuration.

HCDT
Hold current delay time in increments of 1 millisecond.

AIN
Causes a read of Analog I/O channel <chan> . Data is saved to variable <var>.

JSC
Joystick center position. Updated by IJSC command or directly as shown.

JSDB
Joystick deadband. Updated by IJSC command or directly as shown.

JSFS
Joystick full scale. Updated by IJSC command or directly as shown.

MAC
Motor acceleration current setting in percent. Range is 0 to 100. This setting is used whenever velocity is changing. Factory default is 25.

MRC
Motor run (slew) current setting in percent. Range is 0 to 100. Factory default is 25.

MHC
Motor hold current setting in percent. Range is 0 to 100. This setting us used HCDT milliseconds after motion stops. Factory default is 5.

MSEL
Microstep resolution setting. Valid <param> settings are: 2, 4, 8, 16, 32, 64, 128, 256, 5, 10, 25, 50, 125, 250. Factory default is 256.

POSCAP
The value of the position counter at the time of a TRIP on Input/Position/Time/Velocity.

PMHCC
Position maintenance hold current change – 0 to MHC.

STLDM
Stall Detect Mode Setting. Valid "num" settings are: 0 and 1:

0 = Stop Motor when detecting a stall.
1 = Don't stop motor when stall is detected.

New Flags

DRVEN
Drive enable flag. Setting DRVEN = 1 enables the drive output. Factory default is 1.

DRVRS
Drive reset flag. Setting DRVRS = 1 resets the drive to phase B on fullstep. Factory default is 0.

New Math/Logic Functions

FRC
Returns the fractional part of a floating point number.

INT
Returns the integer part of a floating point number.

SECTION 1

The LYNX Terminal Software

Section Overview

This section will cover the usage of the LYNX Terminal software, which is included with your LYNX family product. There are two main benefits to be gained by using this software: First and most importantly, it includes the upgrade utility which allows you to install software upgrades to your LYNX product. The LYNX software cannot be upgraded without this utility! Second, it features two basic windows: A text editor window for writing programs, and a terminal window for communicating with your LYNX system. The multiple document interface allows you to have multiple windows, both editor and/or terminal, open at the same time. Each terminal window can have it's preferences configured independantly in case you have more than one LYNX product connected to different COM ports on your PC. This program also eliminates the need to use two separate programs, for example Notepad and HyperTerminal, to program your system. Covered in this section are:

- Installation and Setup.
- The Menu Command Structure.
- Using the LYNX Terminal to Program.
- Upgrading the LYNX Software.

Installation and Setup

System Requirements

- IBM Compatible 486 or higher PC.
- Windows 95/98 or Windows NT4.0 or 2000
- 5 MB hard drive space.
- A free serial communications port.

Installation

The LYNX Terminal software is a programming/communications interface. This program was created by IMS to simplify programming and upgrading the MicroLYNX. The LYNX Terminal is also necessary to upgrade the software in your MicroLYNX. These updates will be posted to the IMS website at www.imshome.com as they are made available.

To install the LYNX Terminal to your hard drive, insert the CD into your CD-ROM Drive. The 3.5" CD, while smaller than typical compact disks, will work in any horizontally mounted, tray-type CD drive.
To start the installation click "Start > Run" and type "[Drive Letter]:\LYNX terminal\IMS LYNXTerminal.exe" in the "Open" box.

Follow the on-screen instructions to complete the installation.
Detailed instructions for the IMS LYNX Terminal software can be located in The LYNX / MicroLYNX Software Reference Manual.

1) To open the LYNX Terminal select Start > Programs > Lynx_Terminal > Lynx_Terminal.
2) Click the File Menu Item "Edit>Preferences".
3) Click the "Comm Settings" tab.
4) Select the Communications Port that you will be using with your MicroLYNX.

5) The BAUD rate is already set to the MicroLYNX default. Do not change this setting until you have established communications with the MicroLYNX System.

6) The "Window Size" settings are strictly optional. You may set these to whatever size is comfortable to you.

7) Click "OK". The settings will be automatically saved upon a normal shutdown.

8) Apply power to the MicroLYNX System. The following sign-on message should appear in the terminal window:

```
Program Copyright © 1996-2000 by:
Intelligent Motion Systems, Inc.
Marlborough, CT 06447
VER = 1.300     SER =XXXXXXXXX
```

If you can see this sign-on message then you are up and running! If the sign-on banner does not appear, try using a software reset: hold down the "Ctrl" key and press "C" (^C). If the sign-on banner still doesn't appear then there may be a problem with either the hardware or software configuration of the MicroLYNX or Host PC.

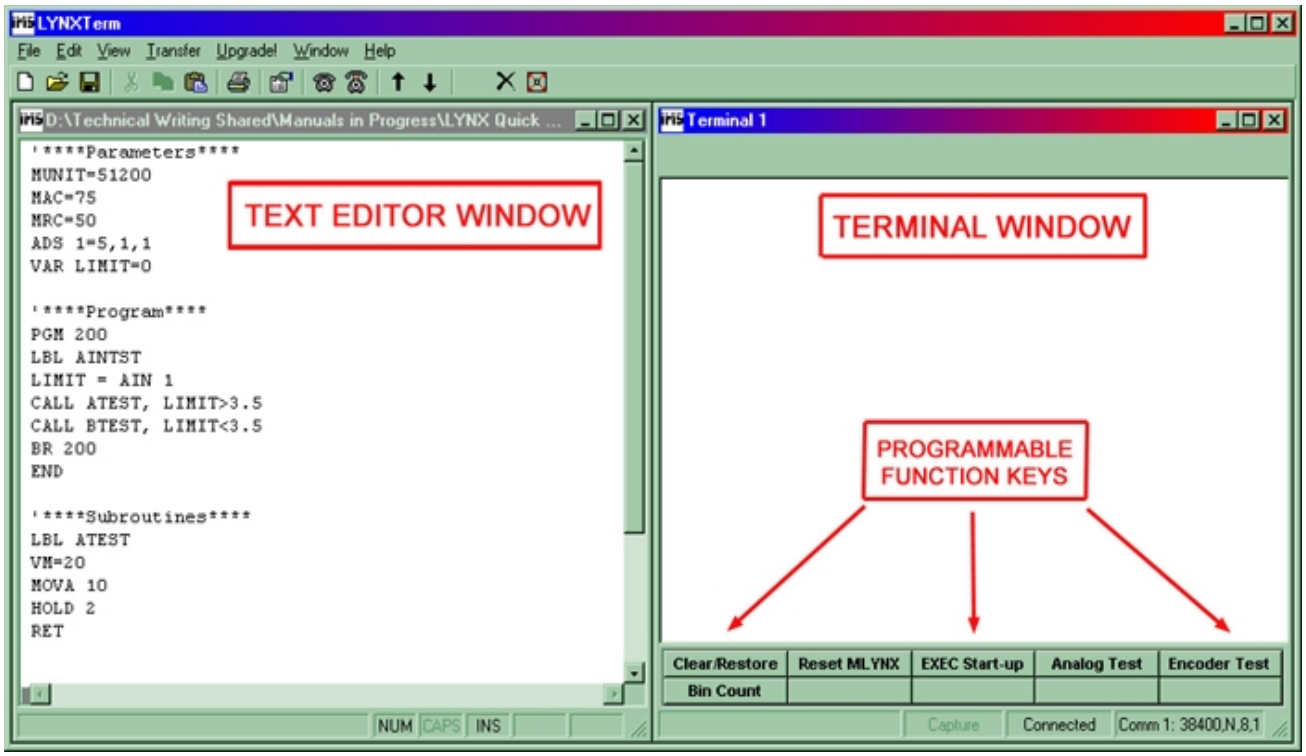


Figure 1.1: The LYNX Terminal Main Screen

Using the LYNX Terminal Software

The LYNX Terminal software is an easy to setup and use interface for MicroLYNX programming. It is also required to upgrade the software in the MicroLYNX. The LYNX Terminal program is fully covered in the LYNX Product Family Operating Instructions. Its coverage in this document is limited to what is required to communicate with the MicroLYNX, and to create, edit and download MicroLYNX programs.

Configuring Communication Settings

The communications settings are configured by means of the "Preferences Dialog". This dialog is accessed through the "Edit > Preferences" menu item or by clicking the "Preferences" icon on the toolbar. The preferences dialog gives the user the ability to set the format for text size, font and color, as well as general communications settings. It is set by default to the optimum communications settings for the MicroLYNX. If

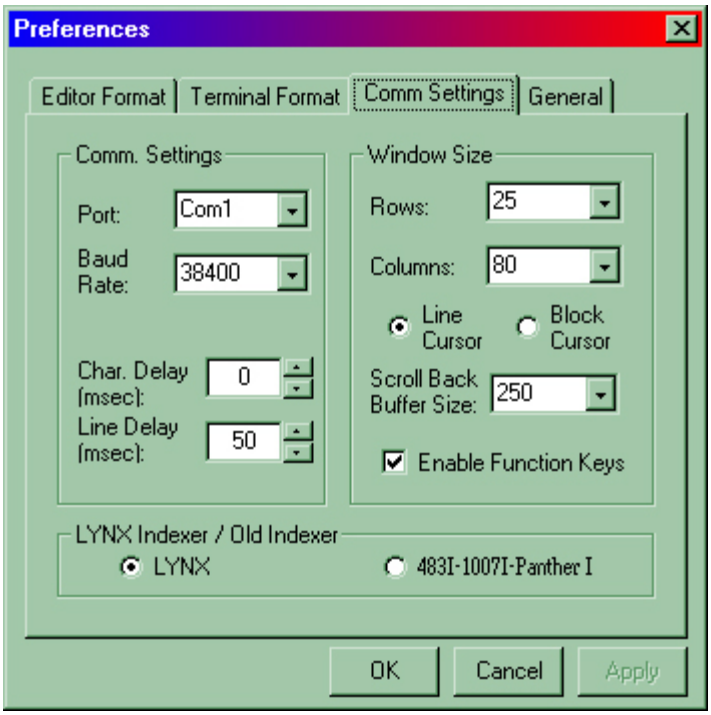


Figure 1.2: LYNX Terminal Communications Settings

you change the BAUD rate setting for the MicroLYNX, power will have to be cycled for the change to take effect. Ensure that the LYNX Terminal preferences are adjusted for the new BAUD settings.

Downloading a Program to the MicroLYNX

There are two ways to download programs to the MicroLYNX:

1] Directly from the text editor window of the LYNX Terminal.
2] From a text file located on a hard drive or removeable disk.

To download a program from the text editor window click the menu item "Transfer > Download". The dialog shown in Figure 1.3 will open. Select the "Source Type > Edit Window" option, click download. The program will transfer to the MicroLYNX.

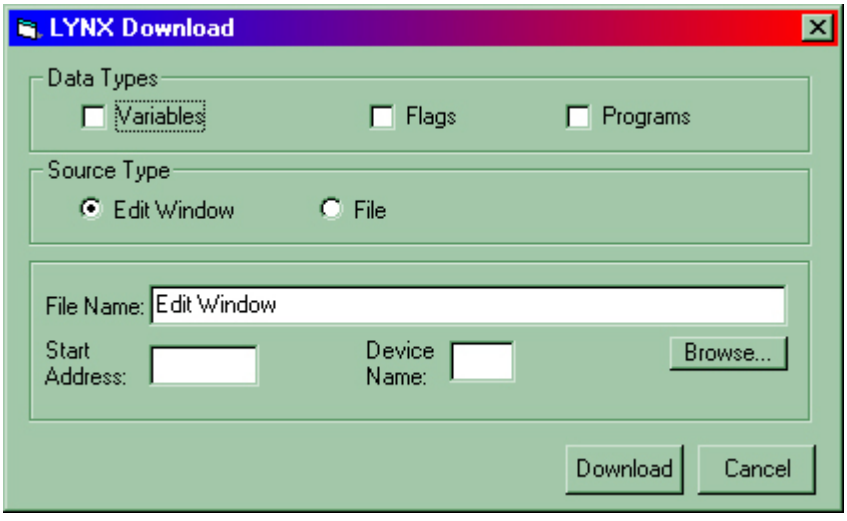


Figure 1.3: LYNX Terminal Download Dialog

Programs can be downloaded to the MicroLYNX from a text file by selecting “Source Type > File” on the dialog and typing in a drive location:\file name in the “File Name” box on the dialog, or browsing to the file location.

Uploading a Program From the MicroLYNX

Programs may also be uploaded from the MicroLYNX in two ways:

1] Directly to the text editor window of the LYNX Terminal.
2] To a text file located on a hard drive or removeable disk.

To upload a program to the text editor window click the menu item “Transfer > Upload”. The dialog shown in Figure 1.4 will open. Select the “Destination Type > Edit Window” option, click “Upload. The program will transfer from the MicroLYNX.

Programs may be uploaded from the MicroLYNX to a text file by selecting “Destination Type > File” on the dialog and typing in a drive location:\file name in the “File Name” box on the dialog.

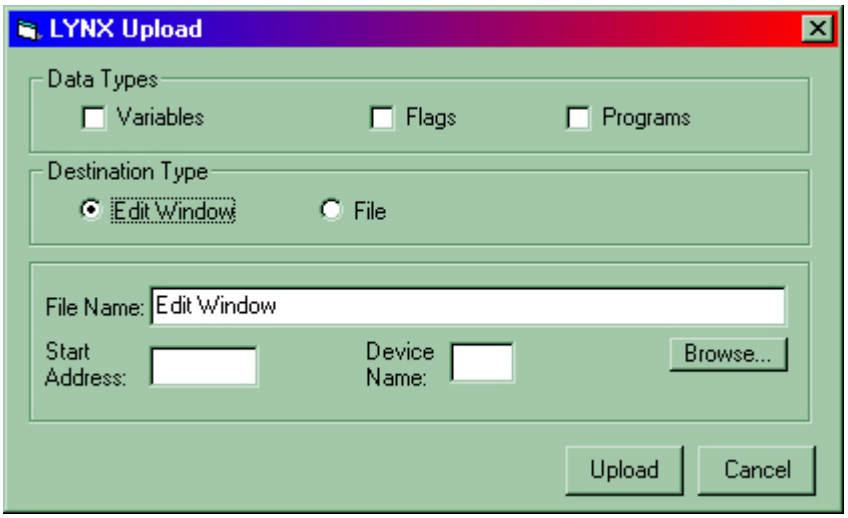


Figure 1.4: LYNX Terminal Upload Dialog

Setting the Programmable Function Keys

The LYNX Terminal features the capability of programming up to 10 function keys, a feature found only on more advanced terminal programs. These can be set to provide quick acces s to commonly used LYNX Immediate mode commands, execute programs, or even hold entire LYNX programs as there is no character limit for each function.

A fly-out dialog can be brought up by clicking the arrow on the right of the function key “Contents” field (see figure 1.5 on the following page) which enables the programmer to embed common ASCII control codes in the function key text string.

To access the function key setup dialog, right-click the function key area on the terminal window.

To setup the function keys:

1] Enter a caption in the “Caption” text field, this will be displayed on the function button.
2] Enter the text string consisting of LYNX commands and ASCII control codes. Remember to terminate each command with a line feed (^M) and and appropriate pause time (typically 1 sec @ 9600bps, or ^p)
3] Click “Done” to set the function.

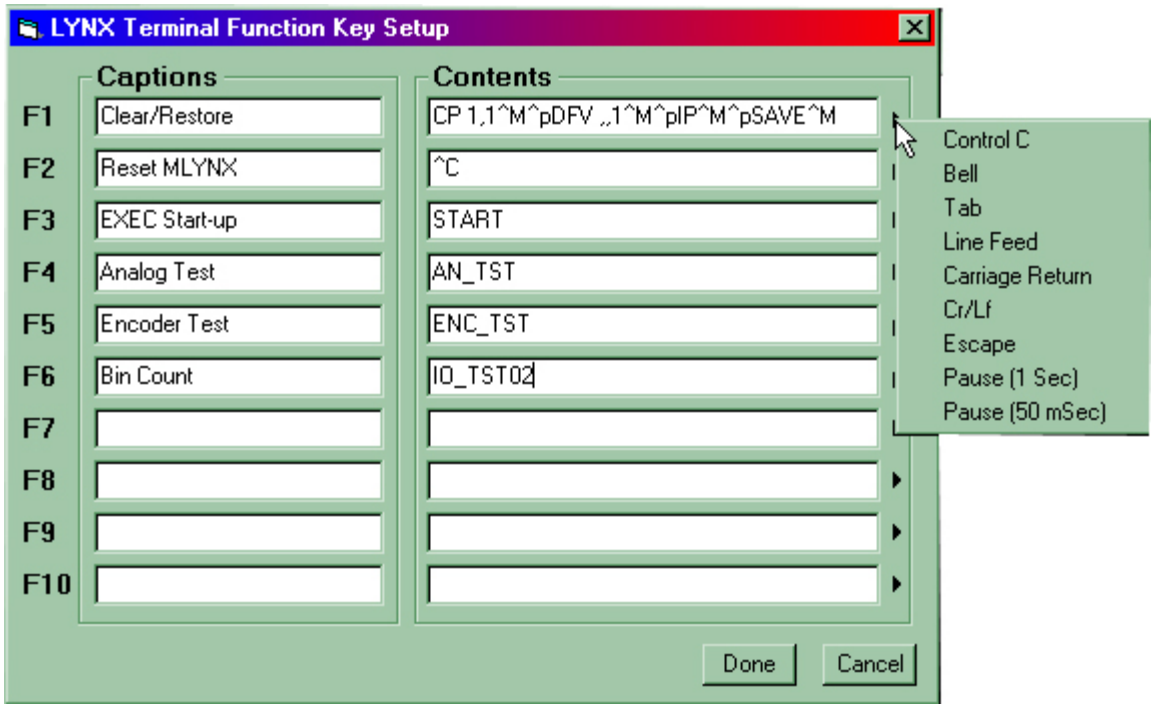


Figure 1.5: LYNX Terminal Function Key Setup

Upgrading the Firmware in Your LYNX Product

1] With power disconnected to the LYNX Product, put the upgrade switch in the ON position.
2] Connect power to the LYNX Product.
3] Open the LYNX Terminal Software.
4] Select the Terminal screen.
5] Click the "Upgrade" menu item on the Menu bar.
6] Follow the instructions.
7] Put the upgrade switch in the OFF position and cycle power.

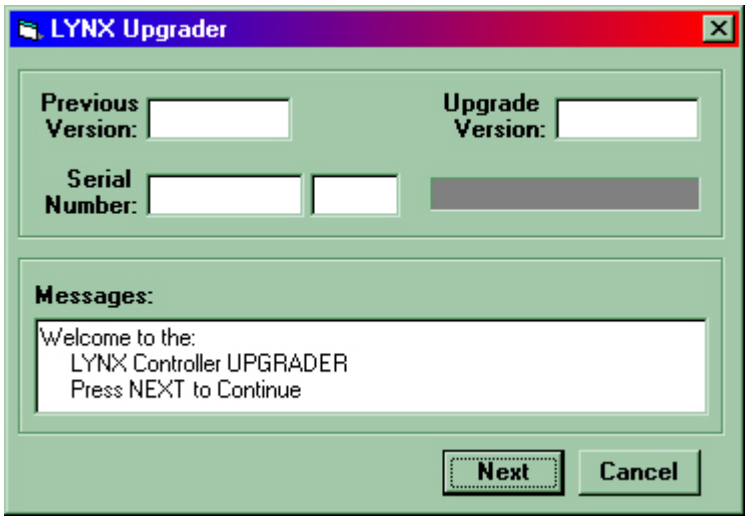


Figure 1.5: LYNX Terminal Upgrader Utility Dialog

SECTION 2

Introduction to LYNX Programming

Section Overview

This section will cover the tools required to effectively program the LYNX product, as well as the basic components of the Lynx Software, then it will cover in depth the most commonly used commands and variables. The LYNX instruction set features a large arsenal of commands which allows it to be very flexible to what applications it is used for, however, the basic commands will apply to most programs. Section 4 of this document, *LYNX Programming Language Reference*, contains detailed descriptions of each instruction, variable, flag and keyword, as well as real-world usage examples for each.

Throughout this section, there a few things for you to take note of:

The word "True" and the number "1" are used interchangeably, as are "False" and "0". These refer to digital logic states. True will ALWAYS be equal to 1, False will ALWAYS equal 0.

The apostrophe character (') is recognized by the LYNX as a comment character. Any text in a program that follows an apostrophe will not be loaded into user memory space. It is a good practice to comment your programs as you are learning the LYNX Programming Language. This will be valuable in debugging your program as it will provide a step-by-step description of each program step. Below is a sample line of commented LYNX code:

```
ACCL=360            'Set the acceleration variable to 360 munits per second²
```


Tools Required:

Terminal

The terminal can be at a minimum a hand held terminal or a DOS driven terminal such as Pro Comm Plus. IMS recommends that the LYNX Terminal software produced by IMS be used, however, either Terminal (Windows 3.1x) or HyperTerminal (Windows 95/98) can be used if you are unable to use the LYNX Terminal. Terminal can be located in *Program Manager/Accessories/Terminal* for Window 3.1x. HyperTerminal for Windows 95/98 can be found in *Programs/Accessories/HyperTerminal*.
The settings (whichever terminal is used), will be: ANSI Terminal, Direct connect to COM port, BAUD Rate = 9600, Data Bits = 8, Parity = None, Stop Bits = 1, Flow Control = NONE.

TIP: The terminal that is included with Windows 3.1x features programmable function keys which can be configured for the commands that you commonly will use (i.e. CP 1,1, IP, DVF). If you are using the upgrade version of Windows 95/98 and have upgraded from Windows 3.1x, the executable file should still be located at c:\windows\terminal.exe.

NOTE: Here is a known bug with HyperTerminal: If the horizontal scroll bar is not set all the way to the bottom left of the window, the commands issued to the LYNX may appear garbled. This is corrected by dragging the scroll bar all the way to the left.

Text Editor

A text editor is recommended for writing and editing the programs. The program then can be simply saved then uploaded as a text transfer with the *Transfer-Send Text file*. The figure on the previous page illustrates the most effective screen setup for using HyperTerminal together with the Windows 95/98 text editor, Notepad, . Notepad is located at *Start-Programs-Accessories-Notepad* for Windows 95/98, and in the program group *Accessories* in the Windows 3.1x program manager.

Basic Components of LYNX Software

Instructions

An instruction results in an action, there are three types:

Motion

Motion instructions are those that result in the movement of a motor. The syntax of these commands are as such: first type the command followed by a space, and then the velocity or position data. For example, *MOVA 2000* will move the motor to position 2000.

I/O

An I/O instruction results in the change of parameters or the state of an Input or Output. The syntax of these commands are as such: first type the command followed by a space, then the I/O #, then an equal sign, then the data. Example: *IO 21=1* will set I/O 21 true.

Program

A program instruction allows program manipulation. The syntax of these vary due to the nature of the command. Some examples would be as such: *PGM 100*, this command toggles the system into program mode starting at address 100. *BR Loop, IO 21=1*, this command will Branch to a program labeled Loop if I/O 21 is true.

System

A system instruction is an instruction that can only be used in immediate mode to perform a system operation such as program execution (EXEC) or listing the contents of program memory (LIST). For example: EXEC 2000 will execute a program located at line 2000 of program memory space.

Variables

Variables are labeled data that allow the user to define or manipulate data. These can also be used with the built-in math functions to manipulate data. There are two classes of variables, factory defined, and user defined. The syntax for each variable may differ. See Section 4, *LYNX Programming Language Reference*, for usage instructions and examples.

Factory Defined Variables

These variables are predefined at the factory. They cannot be deleted. When a DVF (Delete Variables and Flags) or IP (Initialize Parameters) instruction is given, these variables will be reset to their factory default value. There are two types of factory defined variables. They are:

- Read/Writable: These factory defined variables can have their value altered by the user to effect events inside or outside of a program. For example, ACCL (Acceleration Variable) can be used to set the Acceleration, or POS (Position Variable) can be used to set a position reference point.
- Read Only: These factory defined variables cannot be manipulated by the user, but contain data that can be viewed or used to effect events inside a program. For example, VEL (velocity variable) registers the current velocity of the motor in MUNITs per second. (MUNITs will be explained later in this section.)

User Defined Variables

One of the powerful features of the LYNX is that it allows the user to define variables using the VAR (Variable) Instruction. It is important to note that when a DVF (Delete Variables and Flags) or IP (Initialize Parameters) instruction is given, these variables will be deleted! This class of variable must also be saved to memory using the SAVE instruction or when power is removed or a software reset (^C) occurs they will be lost. There are two types of user defined variables:

- Global Variables: Global variables are variables that are defined outside of a program. The benefit to using a global variable is that no user memory is required. For example, the user can define a variable called SPEED by entering VAR SPEED into the terminal. The user can then set that variable to equal the value of the read only variable VEL (velocity) by entering SPEED = VEL into the terminal.
- Local Variables: This type of user defined variable is defined within a program and can only effect events within that program. It is stored in user memory with the program. Examples of this type of variable will be given later in the section. It is worthy of note that a local variable is not static, but is erased and declared again each time a program is executed.

Flags

Flags show the status of an event or condition. A flag will only have one of two possible states: either 1=true/on/enabled or 0=false/off/disabled. As with variables, there are two classes of flags, factory and user defined.

Factory Defined Flags

Factory defined flags are predefined at the factory and cannot be deleted. When a DVF (Delete Variables and Flags) or IP (Initialize Parameters) instruction is given, these flags will be returned to their factory default state. There are two types of factory defined flags:

- Read/Writable: This type of flag is user alterable. They are typically used to set a condition or mode of operation for the LYNX. For example: RATIOE = 1 would enable ratio mode operation, or EE = 0 would disable the encoder functions.
- Read Only: Read Only flags cannot be modified by the user. They only give an indication of an event or condition. Typically this type of flag would be used in a program in conjunction with the BR (branch instruction to generate an if/then event based upon a condition. For Example: the following line of code in a program BR STOPPROG, ACL = 0 would cause a program to branch to a subroutine named “STOPPROG” when the ACL, the read only acceleration flag, is false.

User Defined Flags

This class of flag is defined by the user by using the instruction FLG. This class of flag can be either contained in a program or defined in immediate mode. There are two types of user defined flags:

- Global Flags: Global flags are flags that are defined outside of a program. The benefit to using a global flag is that no user memory is required. For example, the user can define a flag called IN_POS by entering FLG IN_POS into the terminal.
- Local Flags: This type of user defined flag is defined within a program and can only effect events within that program. It is stored in user memory with the program. It is worthy of note that a local variable is not static, but is erased and declared again each time a program is executed.

Keywords

Keywords are used in conjunction with the PRINT, GET and IP instructions to indicate or control variables and flags. For instance, PRINT UVARS would print the state of all the user-defined variables to the screen. IP FLAGS would restore all the flags to their factory default state.

Most Commonly Used Variables and Commands

Variables:

MUNIT

MUNIT, or motor units, is the scaling function used to put steps into user units.
For example, here is a possible scenario: a ball screw has a 3/8 pitch = .375 inch travel per revolution using a 1.8 degree step motor being stepped by a Half/Full stepper in half step there are 400 steps per revolution. If the user wants to operate in inches, the munit scaler would be:

(1 Rev/.375) X (400 steps/Rev) =400/.375 = 1066.667

type MUNIT=400/.375 *then hit enter*

It is recommended that you allow the LYNX Control Module's math functions to perform all the calculations for you. As in the example, were you to round the result of that calculation to 3 decimal places and enter 1066.667 as the MUNIT, it would lead to positional inaccuracy.

POS

POS indicates the position in munits.

- POS takes its reading from CTR1, which is the counter for Clock 1
- *To read the position, type* PRINT POS *or* PRINT CTR1 *then hit enter*
- *To zero the position, type* POS=0 *then hit enter*

VI

Initial velocity in munits per second.

- *To read the initial velocity, type* PRINT VI *then hit enter*
- *To write to the Initial velocity, type* VI=.25 *then hit enter*

VM

Maximum or final velocity.

- *To read the final velocity, key-in* PRINT VM *then hit enter*
- *To write to the final velocity, key-in* VM=5 *then hit enter*

ACCL

Acceleration in munits per second^2.

- *To read the acceleration, key-in* PRINT ACCL *then hit enter*
- *To write to the acceleration, key-in* ACCL=75 *then hit enter*

DECL

Deceleration in munits per second^2 .

- *To read the deceleration, key-in* PRINT DECL *then hit enter*
- *To write to the deceleration, key-in* DECL=ACCL *then hit enter*

Math Functions

Another powerful feature of the LYNX is its ability to perform common math functions and to use these to manipulate data.

Addition ..... NEW_POS*=POS+CTR3
Subtraction ..... DELTA*=CTR2-POS
Multiplication ..... ACCL=ACCL*2
Division ..... ACCL=ACCL/2
Absolute value ..... WAIT=Abs CTR3
*User-defined variable used as an example.

Motion Commands

MOVA

Move to an absolute position relative to a defined zero position.
For example, type the following commands followed by hitting enter:

```
POS=0
MOVA 200
PRINT POS
```

The terminal screen will read 200

```
MOVA 300
PRINT POS
```

The screen will echo back 300.

MOVR

Move number of steps indicated relative to current position.
For example, type the following commands followed by hitting enter:

```
POS=0
MOVR 200
PRINT POS
```

The terminal screen will read 200

```
MOVR 300
PRINT POS
```

Notice the position echoed is 500 and not 300.

SLEW

Move at a constant velocity.

```
SLEW 2000
```

The motor will move at a constant velocity 2000 munits per second.

HOLD

A HOLD 2 should typically follow any MOVA or MOVR commands in a program so that program execution is suspended until motion is complete.
(Note: There are circumstances where you may not want to hold up program execution.)
Below is a usage example.

```
PGM 1
MOVR 200
HOLD 2
END
PGM
```

I/O Grouping

Group 10
Differential High speed I/O
· I/O Lines 11 – 18
Predefined as differential Step/Direction outputs

Group 20-50
Isolated 5/24vdc I/O
Control Module
· Group 20 = I/O Lines 21 – 26
· Group 30 = I/O Lines 31 – 36
Isolated I/O Module
· Group 40 = I/O Lines 41 – 46
· Group 50 = I/O Lines 51 – 56

IOS

Sets the parameters of the I/O, this command configures the I/O.

Using the PRINT command to read IO parameters
Read all I/O parameters –" PRINT IOS*"*
Read I/O group 20 parameters –" PRINT IOS 20*"*
Read I/O 21 parameters –" PRINT IOS 21*"*

Setting the I/O parameters
Set group 20 I/O parameters – "IOS 20=#,#,#,#,#,#"
Set I/O 25 parameters – "IOS 25=#,#,#,#,#,#"

See the table on the next page for full I/O identification and settings.
For example: To set I/O 25 as a Jog+ input/Low True/Level triggered the following would be entered:
IOS 25 = 16,0,0,0

IO

Used to read/write the binary state of an I/O group or read/write of an individual output. (Note: I/O must be configured as Outputs to set the state of outputs.)

Each I/O Group has 6 weighted bits:
Least Significant Bit is 1, and Most Significant Bit is 6.
Weight of the LSB=1 and MSB=32

*Using the PRINT command to read the state of I/O group 20 - "*PRINT IO 20*"*

To determine the decimal equivalent of the binary state of the whole group, you would add together the decimal weight of each set bit.
Decimal equivalent = 44, because 32 + 8 + 4 = 44

To set the state of I/O group 30 –" IO 30=39*"*
This will set all 6 I/O lines in group 30 to, 100111 the binary equivalent of 39
Decimal equivalent 32 + 4 +2 + 1 = 39

To set the state of individual I/O 31 –" IO 31= 0*"*

The binary equivalent of group 30 is now = 38 = 100110

Decimal equivalent = 38, because 32 + 4 + 2 = 38

*To read state of individual I/O 31 - "*PRINT IO 31*"*

A "1" or "0" will appear (1=true, 0=false)

System Instructions

The following System instructions will be used frequently.

CP

The CP Instruction is used

Program Instructions

PGM

This instruction toggles the LYNX into or out of program mode.

Switch to program mode at address 200	PGM 200
	xxxx
Program starting at address 200	xxxx
	xxxx
Switch out of program mode	PGM

LBL

Assigns a label or name to a program or subroutine.

Switch to program mode at address 200	PGM 200
Label command will name the program	LBL Program1
	xxxx
Program named by lbl command	xxxx
	xxxx
Switch out of program mode	PGM

BR

Used to branch conditionally or unconditionally to a routine.

Switch to program mode at address 200	PGM 200
Label command will name the program	LBL Program1
	xxxx
Program named by LBL command	xxxx
	xxxx
Unconditional branch to Program1	BR Program1
Switch out of program mode	PGM

END

Designates the end of a program.

Switches to program mode at address 200	PGM 200
Label command will name the program	LBL Program1

	xxxx
Program named by LBL command	xxxx
	xxxx
Unconditional branch to Program1	BR Program1
Designates the end of the program	END
Switches out of program mode	PGM

DELAY

Delays program execution in milliseconds.

Switches to program mode at address 200	PGM 200
Label command will name the program	LBL Program1
	xxxx
Program named by LBL command	xxxx
	xxxx
Delay 2 seconds between re-execution of program	DELAY 2000
Unconditional branch to program1	BR Program1
Designates the end of the program	END
Switches out of program mode	PGM

PRINT

Outputs specified text and parameter values to a terminal or terminal software on a Host PC.

Switches to program mode at address 200	PGM 200
Label command will name the program	LBL Program1
	xxxx
Program named by LBL command	xxxx
	xxxx
Prints text in quotes and then POS	PRINT “Position = ” POS
Delay 2 seconds between re-execution of program	DELAY 2000
Unconditional branch to program1	BR Program1
Designates the end of the program	END
Switches out of program mode	PGM

VAR

Command used to define a variable with 8 alphanumeric characters.

Switches to program mode at address 200	PGM 200
Define a variable named Count	VAR Count
Label command will name the program	LBL Program1
	xxxx
Program named by LBL command	xxxx
	xxxx
Prints text in quotes and then POS	PRINT “Position = ” POS
Delay 2 seconds between re-execution of program	DELAY 2000
Unconditional branch to program1	BR Program1
Designates the end of the program	END
Switches out of program mode	PGM

Program mode is the mode that the LYNX must be in to enter programs. This is done by simply typing PGM and then an address between 1 and 8000. After the program has been entered, type PGM to toggle out of program mode.

Check proper hook up of system components to the LYNX Product.

When ready to write a program, it is a good rule of thumb to Clear Program memory with the CP 1,1 command. Delete user-defined Variables and Flags with the DVF command. And Initialize Parameters with the IP command. With the LYNX Product now at factory default, there are no parameters that will throw you off track when and if you need to debug your program.

Solve I/O configuration: is it a clock input or output, is it a user input or a user output, is it a dedicated I/O, is it low true or high true. Configuring the I/O is done using the IOS command.

Compute Scaling factor that scales pulses or steps into user units of degrees, rpm, inches, etc. This is using the MUNIT variable and, if an encoder is installed and enabled, the EUNIT variable also.

Using the text editor, notepad or wordpad, start writing the program. It is often easier to start with the basic motion you want. After verifying that it works, then edit the text file and add the loops and branches as needed.

There are three ways to program the LYNX Product: The first is in immediate program mode where you program as you type. This is not recommended. We recommend using a text editor, using the Copy and Paste functions to simply paste the program onto the LYNX terminal, or using the Send Text file function you can transfer the file to the LYNX terminal.

After the final version of the program has been entered, a SAVE should be issued to save the program from Flash Memory to Non-Volatile Memory.

Program Samples

System Characteristics of Sample Programs

1). The 1.8 degree stepper motor is being driven by an IM483 in 1/256 resolution.
Therefore 1 rev. of the motor is 360/1.8=200; 200 X 256=51200 micro-steps.
The normally open dry contact switch will be between ground and the inputs.
The internal pull-up resistor to 5 VDC for the inputs has been selected by the dip switches.
Therefore when the switch is pressed the input will be grounded or low, and when not pressed it will be 5VDC or high.

2). The 1.8 degree stepper motor is being driven by an IM483 in 1/256 resolution.
One revolution of the motor gives 25 mm of deflection. The normally open dry contact switch will be between ground and the Inputs. The internal pull-up resistor to 5VDC for the inputs has been selected by the dip switches.
Therefore when the switch is pressed the input will be grounded or low, and when not pressed it will be 5VDC or high.

Sample Program 1

1A). This first program will set I/O 21 as an Input to interface a switch. When the Input is pulled low through the switch, the motor will move one revolution. The switch will essentially will initiate the program (G0 Switch).

```
IOS 21=9,0,0,0,0,0   'Set I/O 21 to be a G0 input
PGM 1                'Enter program mode at address 1
LBL InitProg         'Name the following program InitProg
POS=0                'Set position to zero
LBL TurnOnce         'Name the following program TurnOnce
MOVR 51200           'Move relative 51200 steps
HOLD 2               'Suspend program execution until motion has stopped
END                  'Designate the end of the program
PGM                  'Exit program mode
```

1B). The second program will set I/O 21 as an Input to interface a switch. When the Input is pulled low through the switch, the motor will move one revolution. The switch will essentiallyl initiate the program (G0 Switch). Then it will wait 3 seconds, return to zero, and wait for I/O 25 to become true before repeating the cycle. After each cycle it activates one of the 6 LED's until the sixth one is reached then it resets the LED's all off and ends the program.

```
IOS 21=9,0,0,0,0,0   'Set I/O 21 to be a G0 input
IOS 25=0,0,0,0,0,0   'Set I/O 25 to be a User Input, Low true.
PGM 1                'Enter program mode at address 1
LBL InitProg         'Name the following program InitProg
POS=0                'Set position to zero
VAR Lights=1         'Define the variable Lights set it equal to 1
IOS 30=0,1,0,0,0,0   'Set group 30 to all be User Outputs, Low True
IO 30=1              'Set IO group 30 to 1, IO 31 true, Low active
LBL TurnOnce         'Name the following program TurnOnce
IO 30=Lights         'Set IO group 30 all false, Low true, so all high
Lights=Lights*2      'Double the value of Lights( 1,2,4,8,16,32,64)
BR Done, Lights>33   'Conditional Branch to Done if Lights greater than 33
MOVR 51200           'Move relative 51200 steps
HOLD 2               'Suspend program execution until motion has stopped
DELAY 3000           'Delay three seconds
MOVA 0               'Move absolute to zero
HOLD 2               'Suspend program execution until motion has stopped
BR TurnOnce          'Unconditional Branch to TurnOnce
LBL Done             'Name the following program Done
IO 30=0              'Set IO group 30 all false
END                  'Designate the end of the program
PGM                  'Exit program mode
```

Program Sample 2

2A). This program will set I/O 21 & 25 as inputs to interface the switches. When input 21 is true it starts the program which moves the motor at a constant velocity until I/O 25 is true, then it prints its position and returns to zero.

```
IOS 21=9,0,0,0,0,0   'Set I/O 21 to be a Go input
IOS 25=0,0,0,0,0,0   'Set I/O 25 to be a user input
PGM 1                'Enter program mode at address 1
LBL ProgInit         'label the following program ProgInit
POS=0                'Set position to zero
MUNIT=51200/25       'Scale micro-steps into Millimeters
LBL TurnOnce         'label the following program TurnOnce
SLEW 500             'Move at constant velocity of 500 mm per second
```

```
LBL Loop1            'label below program Loop1
DELAY 2              'delay 2 milliseconds
BR Loop1, IO 21=0    'Conditional Branch to Loop1 if io 25 is high
SLEW 0               'Move at constant velocity of 0 mm per second
HOLD 1               'Suspend program execution until motion has stopped
PRINT POS            'Prints position
END                  'Designate the end of the program
PGM                  'Exit program mode
```

2B). This program will run upon power up “Startup”, provided it is saved to NVM prior to power down. The program will first ask for the Speed in mm-per-second at which to slew. Once entered, it will slew at that speed until input 21 is true, then it prints the position where it stopped at returns to zero and asks for another speed.

```
PGM 200              'Enter program mode at address 200
LBL startup          'label the following program Start-up
POS=0                'Set position to zero
MUNIT=51200/25       'Scale micro-steps into Millimeters
IOS 21=0,0,0,0,0,0   'Set I/O 21 to be a Go input
VAR Speed            'define the variable speed
LBL MoveMe           'label the following program MoveMe
PRINT "Enter Speed:" 'Print to terminal Enter Speed:
INPUT Speed          'Allow user to enter data
SLEW Speed           'Move at constant velocity equal to speed
LBL Loop1            'label below program Loop1
DELAY 2              'delay 2 milliseconds
BR Loop1, IO 21=0    'Conditional Branch to Loop1 if io 21 is high
PRINT POS            'Prints position
MOVA 0               'Move back to zero position
HOLD 2               'Suspend program execution until motion has stopped
BR Moveme            'Unconditional Branch to MoveMe
END                  'Designate the end of the program
PGM                  'Exit program mode
```

Cut to Length Application

This program asks for several variables using the Print and Input commands. It then will start feeding the material in the cutsize increments with a delay adjustable by the encoder input on Counter 3. When the material leftover is less than the cutsize, the user has the option to modify the cutsize. When there is no material left it will exit the program.

```
PGM 1                                 'Start program mode at address 1
LBL Cutstuff                          'Name the program "Cutstuff"
MUNIT=51200/25                        'Scale steps into user units
CTR3=100                              'Set Counter 3 (Clock 3 counter) to 100
POS=0                                 'Set Position Register (Clock 1 counter) to 0
VAR Feedrate=0                        'Define the Variable "Feedrate" and set it to 0
VAR Cutsize=0                         'Define the Variable "Cutsize" and set it to 0
VAR Length=0                          'Define the Variable "Length" and set it to 0
VAR Leftover=0                        'Define the Variable "Leftover" and set it to 0
VAR Enter=0                           'Define the Variable "Enter" and set it to 0
VAR Time=100                          'Define the Variable "Time" and set it to 100
FLG Answer1=0                         'Define the Flag  "Answer1" and set it to 0
PRINT "Enter Feed Rate in inches/sec- "; 'Prompts user for Feed Rate speed
INPUT Feedrate                        'Enters the Data entered by the user into Feedrate
                                      'variable
PRINT "Enter length of raw material in inches -";      'Prompts user for Length
INPUT Length                          'Enters the Data entered by user into
                                      'Length variable
LBL Cutting                           'Name the program "Cutting"
PRINT "Enter in inches, length of Cut -";'Prompts user for length of Cut
INPUT Cutsize                         'Enters the Data entered by user into Cutsize variable
LBL Go_now                                  'Name the program "go_now"
Leftover=Length-POS                         'Set Leftover equal to material length less
                                            'already cut
BR Toosmall, Leftover<Cutsize               'Branch to "toosmall" if leftover is less than
                                            'cutsize
Time=abs CTR3                               'Set time equal to the absolute value of ctr3
DELAY Time                                  'Delay for (time * .001) seconds
VI=Feedrate/100                             'Set Initial Velocity to Feed rate divided by 100
VM=Feedrate                                 'Set Max Velocity to Feed rate entered
MOVR Cutsize                                'Move number of inches entered for Cutsize
HOLD 2                                      'Suspend the Program execution until Motion stops
BR Done, POS=Length                         'Branch to "Done" if amount cut is equal to
                                            'Length
BR Done, POS>Length                         'Branch to "Done" if amount cut is less than
                                            'Length
BR Go_now                                   'Branch to "Go_now"
LBL Toosmall                                'Name the program "Toosmall"
BR Done, Leftover=0                         'Branch to "Done" if remaining material is equal
                                            'to 0
PRINT " Remaining material is smaller or equal to cutsize!!"
PRINT "You have " leftover " inches remaining  "
PRINT  "======================================="
PRINT " Do you wish to modify cutsize: Yes=1, No=0 "
INPUT Answer1                               'Enters the Data entered by user into answer1
                                            'Flag
BR Cutting, Answer1=1                       'Branch to "Cutting" if user wants to modify
                                            'cutsize
LBL Done                                    'Name the program "Done"
PRINT " Program has ended.   Remove all debris !!"
PRINT  "======================================="
PRINT "To run program again type cutstuff"
END                                         'End of Program
PGM                                         'Ends Program Mode
```

SECTION 3

Functional Grouping of the Instruction Set

Section Overview

This section covers contains a logical grouping of the LYNX product family instruction set. Each subsection contains a tableized summary displaying a description, usage example and default setting for each instruction, variable, flag or keyword. In the case where a command can logically be placed in more than one group, it is duplicated in each group. The following functional groups are presented:

- Acceleration and Deceleration
- Position
- Encoder
- Miscellaneous Motion
- Event
- Instructions (Immediate Mode)
- Miscellaneous and Setup Flags
- Velocity
- Drive and Motor
- I/O
- Data
- Instructions (Program Mode)
- Miscellaneous and Setup Variables
- Mathematical and Logical Functions

Using the Tables

The instruction set summary tables are set up in the manner illustrated in the following example:

Example Table			
Command	**Usage Example**	**Description**	**Default**
INST	INST	This instruction causes that event.	
VAR	VAR=<num>,<mode>	Variable contains some data. <num>= Some number (range) or unit of measure. <mode> = 0: Does this. <mode> = 1: Does that.	<num> = 1024 <mode> = 0
FLAG	FLAG=<flg>	Flag enables/disables some function. <flg> = 0: Disabled. <flg> = 1: Enabled.	0

Command

The command is given in the left hand colunm. The Adobe Portable Document format (*.pdf) version of this manual has hyperlinks built-in to allow for easy linking from one portion to another. By clicking on the command in the command column, the user is linked to the full description of the command in the Language Reference section of this document.

Usage Example

The usage example column illustrates how the instruction, variable or flag would be used in a program or in immediate mode. In the case of the expressions bracketed by the <> symbol only the contents would be typed not the symbols themselves. For example: VAR=<num>,<mode> would be entered VAR=23, 1 *(arbitrary numbers used in example)*. The following codes are mostly self explanatory and are used to identify the various settings:

<num>	=	Some number.
<param>	=	Parameter.
<time>	=	Time.
<flg>	=	Flag, this will be 1 or 0.
<percent>	=	Percentage.
<lbl/addr>	=	Program label or address.
<mode>	=	Mode.
<chan>	=	Channel.
<func>	=	Function.
<cond>	=	Condition.
<state>	=	Logic state.

Description

The description column contains a brief description of the command and an elaboration of the expression bracketed by the <> symbols.

Factory Default

This column contains the factory default setting of the variable or flag discussed.

Acceleration and Deceleration

Acceleration and Deceleration Related Variables and Flags			
Command	**Usage Example**	**Description**	**Default**
ACCL	ACCL=<num>	Peak acceleration value. <num>= User units/sec^2.	1000000.000
ACL	BR <lbl/addr>, ACL=<flg> PRINT ACL	Read-only acceleration flag. <lbl/addr> = Program label or address. <flg>=1: Accelerating. <flg>=0: Not accelerating.	0
ACLT	ACLT=<param>	Acceleration type variable. <param>=0: User Defined. <param>=1: Linear. <param>=2: Triangle S-Curve. <param>=3: Parabolic. <param>=4: Sinusoidal S-Curve.	1
ACLTBL	ACLTBL=<num>, <val>	User-defined acceleration profile table. <num>=0 - 256 <val>=0.00-1.00	Empty
DCL	BR<lbl/addr>, DCL=<flg> PRINT DCL	Read-only deceleration flag. <lbl/addr> = Program label or address. <flg>=1: decelerating. <flg>=0: not decelerating.	0
DCLT	DCLT=<param>	Deceleration type variable. <param>=0: User Defined. <param>=1: Linear. <param>=2: Triangle S-Curve. <param>=3: Parabolic. <param>=4: Sinusoidal S-Curve.	1
LDCLT	LDCLT=<param>	Specifies deceleration type used when a limit is reached. <param>=0: User Defined. <param>=1: Linear. <param>=2: Triangle S-Curve. <param>=3: Parabolic. <param>=4: Sinusoidal S-Curve.	1
LDECL	LDECL=<num>	Peak deceleration value when stopping due to a limit. <num>= user units/sec^2.	1000000.000

Velocity

Velocity Related Variables, Flags and Instructions			
Command	**Usage Example**	**Description**	**Default**
JOGS	JOGS=<num>	Jog speed variable. <num>= user units/sec.	256000.000
PMV	PMV=<num>	Position maintenance velocity variable. <num>= user units/sec.	10240.000
SLEW	SLEW <num>=<mode>	Slew the motor at a constant velocity instruction. <vel> = user units/sec. <mode> = 0: Use acceleration ramp. <mode> = 1: Do not use acceleration ramp.	Mode 0 used if <mode> not specified.
SSTP	SSTP<mode>	Stop the current motion using the specified deceleration profile and optionally stop the program. <mode> = 0: Stop motion only. <mode> = 1: Stop motion and program.	Mode 0 used if <mode> not specified.
VCHG	BR<lbl/addr>, VCHG=<flg> PRINT VCHG	Read-only flag indicates when velocity is changing. <lbl/addr> = Label or address of program. <flg> = 0: Velocity is not changing. <flg> = 1: Velocity is changing	0
VEL	BR<lbl/addr>, VEL=<num> PRINT VEL	Register that contains the actual velocity of the axis. In user units per second. Read-only. <lbl/addr> = Label or address of program. <num> = user unit/sec.	0.000
VI	VI=<num>	Initial velocity of the axis during a point-to-point motion. <num> = user units/sec	102400.00
VM	VM=<num>	Maximum velocity reached by the axis during a point-to-point motion. <num> = user units/sec.	768000.000

Position

Position Related Variables, Flags and Instructions			
Command	**Usage Example**	**Description**	**Default**
FIOS	FIOS<num1>,<num2>,<line>	Find I/O switch instruction. Parameters are optional. <num1> = ± speed in user units/sec. <num2> = ± creep in user units/sec. <line> = I/O line.	If not specified: <num> = VM <param> = VI
MOVA	MOVA <num>, <mode>	Perform point-to-point move or index to an absolute position instruction. Use of <mode> is optional. <num> = Absolute position. <mode> = 0: Motion ceases when position is reached. <mode> = 1: Motion part of a profile, does not decelerate.	Mode 0 is used when mode not specified.
MOVR	MOVR <num>, <mode>	Perform point-to-point move or index to a relative position instruction. <num> = Absolute position. <mode> = 0: Motion ceases when position is reached. <mode> = 1: Motion part of a profile, does not decelerate.	Mode 0 is used when mode not specified.

Position Related Variables, Flags and Instructions cont'd			
Command	**Usage Example**	**Description**	**Default**
MVG	BR <lbl/addr>, MVG=<flg> PRINT MVG	Read-only flag indicates when motion is taking place. <lbl/addr> = Program label or address. <flg> = 0: Axis is stationary. <flg> = 1: Axis is in motion.	0
PCHG	BR <lbl/addr>, PCHG=<flg> PRINT PCHG	Read-only flag indicates when the axis is trying to reach a specified relative or absolute position. <lbl/addr> = Program label or address. <flg> = 0: Axis is moving in a "jog" or "slew". <flg> = 1: Axis isindexing to a position.	0
POS	POS=<num> PRINT POS	Register which contains the axis position in user units. <num> = ± Position	0.000
POSCAP	PRINT POSCAP	Read-only variable: position at time of TRIP.	0.000

Drive and Motor

Drive and Motor Related Variables, Flags and Instructions			
Command	**Usage Example**	**Description**	**Default**
DRVEN	DRVEN=<flg>	Drive enable flag: enables/disables drive output. <flg> = 0: Drive output disabled. <flg> = 1: Drive output enabled.	1
DRVRS	DRVRS=<flg>	Drive reset flag: resets drive output. <flg> = 0: Drive not reset. <flg> = 1: resets the drive to phase B on fullstep.	0
DRVTP	PRINT DRVTP	Read-only drive type variable. Provides a means to interrogate system to determine the type of drive. Response = 2: IM483H Response = 4: IM805H	
HCDT	HCDT=<time>	Holding current delay time variable. <time> = Time in milliseconds.	0
MAC	MAC=<percent>	Motor acceleration current variable. Used when velocity is changing. <percent> = 0 - 100	25
MHC	MHC=<percent>	Motor holding current variable. Used when axis is stationary. <percent> = 0 - 100	5
MRC	MRC=<percent>	Motor run current variable. Used when axis is at max velocity. <percent> = 0 - 100	25
MSEL	MSEL=<param>	Microstep resolution variable. Valid <param> settings are: 2, 4, 8, 16, 32, 64, 128, 256, 5, 10, 25, 50,125, 250.	256
MSDT	MSDT=<time>	Motor settling delay time variable. <time> = 0 - 65,535 milliseconds.	0
MUNIT	MUNIT=<num>	Motor units variable specifies the number of clock pulses per user unit. <num> = Clock pulses/user unit	1.000
PMHCC	PMHCC=<percent>	Variable specifes the position manitenance hold current change. <percent> = 0 to MHC	0

Encoder

Encoder Related Variables, Flags and Instructions			
Command	**Usage Example**	**Description**	**Default**
EDB	EDB=<num>	Encoder deadband variable specifies the ± length of the deadband for position maintenance. <num> = user units.	2.000
EE	EE=<flg>	Master enable flag for all encoder functions. <flg> = 0: Encoder functions disabled. <flg> = 1: Encoder functions enabled.	0
EUNIT	EUNIT=<num>	Conversion variable for converting motor steps or user units to encoder counts. <num> = encoder counts per user unit.	1.000
PME	PME=<flg>	Position maintenance enable flag. <flg> = 0: Position maintenance disabled. <flg> = 1: Position maintenance enabled.	0
PMV	PMV=<num>	Position maintenance velocity variable. <num> = user units/sec.	10240.000
STALL	BR<lbl/addr>, STALL=<flg> PRINT STALL	Flag which indicates if the motor has stalled. <lbl/addr> = Program label or address. <flg> = 0: Axis not stalled. <flg> = 1: Axis stalled.	0
STLDE	STLDE=<flg>	Flag enables stall detection/ <flg> = 0: Stall detection disabled. <flg> = 1: Stall detection enabled.	0
STLDM	STLDM=<mode>	Stall detect mode setting determins whether motor stops when a stall is detected. <mode> = 0: Stop motor. <mode> = 1: Do not stop motor.	0
STLF	STLF=<num>	Stall factor variable. <num> = User units.	10

I/O

I/0 Related Variables, Flags and Instructions			
Command	**Usage Example**	**Description**	**Default**
ADS	ADS<chan>=<aunit>,<func>,<law>	Setup variable for the Analog Input/Joystick module. <chan> = Channel # (1 or 2). <aunit> = User Unit = MUNIT * AUNIT. <func> = 1: Analog input. <func> = 2: Joystick interface. <law> = 1: Linear. <law> = 2: Square law. <law> = 3: Cube law. <law> adjusts the joystick position to motor velocity transformation.	1, 1, 1
AIN	<var>=AIN<chan>	Variable causes a read of analog input channel. <var> = Variable to which data is saved. <chan> = Analog input channel.	

I/0 Related Variables , Flags and Instructions cont'd			
Command	**Usage Example**	**Description**	**Default**
IJSC	IJSC	Instruction supports the Analog Input/Joystick Interface in joystick mode.	
IO	IO<line>=<lstate> IO<group>=<bstate>	Variable reads or writes the state of an I/O <line> or <group>. <lstate> = 0: I/O line inactive. <lstate> = 1: I/O line active. <bstate> = (0-63): Binary state of all lines in group.	
IOF	IOF<group>=<param>	Variable sets the level of digital filtering to be applied to a specified I/O Group. <group> = 10 - 50 <param> = 1 - 7	Group 10 = 0 Groups 20-50 = 7
IOS	See Language Reference for usage example.	Variable configures the I/O, also used as a keyword with the IP instruction.	
JSC	JSC=<num>	Joystick center position variable. Automatically updated by IJSC or manually using: <num> = 0 - 4095 (AUNIT = 1)	2048
JSDB	JSDB=<num>	Joystick deaband variable. Automatically updated by IJSC or manually using: <num> = 0 - 4095 (AUNIT = 1)	10
JSE	JSE=<flg>	Joystick Enable/Disable Flag. Enables velocity mode for the Analog Input/Joystick Module. <flg> = 0: Disabled <flg> = 1: Enabled	0
JSFS	JSFS=<num>	Joystick full scale variable. Automatically updated by IJSC or manually using: <num> = 0 - 4095 (AUNIT = 1)	2038
LIMSTP	LIMSTP=<flg>	Flag specifies whether or not motion will cease when a limit is reached. <flg> = 0: Motion will not stop. <flg> = 1: Motion will stop.	1

Miscellaneous Motion

Miscellaneous Motion Related Variables, Flags and Instructions			
Command	**Usage Example**	**Description**	**Default**
BLE	BLE=<flg>	Flag enables/disables backlash compensation. <flg> = 0: Disabled. <flg> = 1: Enabled.	0
BLM	BLM=<mode>	Variable specifies the mode for backlash compensation. <mode> = 0: Mathematical Compensation. <mode> = 1: Mechanical Compensation.	Mode 0
BLSH	BLSH=<num>	Backlash compensation amount. <num> = User Units.	0.000
CTR1		Counter which represents the raw counts sent to the primary motor.	0.000
CTR2		Counter which represents the raw counts received from the encoder.	0.000
CTR3		Counter which represents the raw counts of the clock seen on I/O 15 and 16.	0.000
HAE	HAE=<flg>	Flag which enables/disables half axis scaling mode. <flg> = 0: Disabled. <flg> = 1: Enabled.	0

Miscellaneous Motion Related Variables, Flags and Instructions cont'd			
Command	**Usage Example**	**Description**	**Default**
HAS	HAS=<param>	Variable defines the scaling factor for half axis mode. <param> = -1 to <1	0.000
RATIO	RATIO=<param>	Variable sets the ratio for a secondary drive to a specified value. <param> = -2 to <2	1.000
RATIOE	RATIOE=<flg>	Master flag enables/disables ratio functions. <flg> = 0: Disabled. <flg> = 1: Enabled.	0
RATIOW	RATIOW=<num>	Pulse width for the secondary channel(s) being used to drive the motor(s) in ratio mode. <num> = 0 - 254 (1 - 254 = 50ns increments)	0
STEPW	STEPW=<num>	Pulse width for the step clock of the primary axis. <num> = 0 - 254 (1 - 254 = 50ns increments)	0

Data

Data Related Instructions, Keywords and Flags			
Command	**Usage Example**	**Description**	**Default**
ALL	PRINT ALL IP ALL GET ALL	Keyword used with the GET, PRINT, and IP instructions to indicate the inclusion of all variables, flags and program space where applicable.	
CPL	CPL <var/flg>	Instruction that performs the two's complement of the specified variable or flag. <var/flg> = Variable or flag.	
DEC	DEC <var>	Instruction used to decrement the specified variable by 1. <var> = Variable.	
DVF	DVF <param1>,<param2>	Instruction that deletes user-defined variables and flags. <param1> = 0: All user vars and flags deleted. <param1> = 1: Only user vars deleted. <param1> = 2: Only user flags deleted. <param2> = 0: All global and local user vars and/or flags deleted. <param2> = 1: Only global user vars and/or flags deleted. <param2> = 2: Only local user vars and/or flags deleted.	If no parameter is specified, both will be 0
ERR	PRINT ERR	Read-only status flag indicates whether an error has occurred.	
ERRA	PRINT ERRA	Read-only variable displays the memory location that the error occurred.	
ERROR	PRINT ERROR	Read-only variable contains the error code for the most recent error. See Appendix B for a list of possible errors.	
FAULT	FAULT=<flg>	Flag allows the user to enable/disable the fault indicator LED. <flg> = 0: Disabled. <flg> = 1: Enabled.	1
FLAGS	PRINT FLAGS IP FLAGS GET FLAGS	Keyword used with the GET, PRINT, and IP instructions to indicate the inclusion of only flags.	

Data Related Instructions, Keywords and Flags cont'd			
Command	**Usage Example**	**Description**	**Default**
FLG	FLG <name>=<state>	Instruction to define a user flag that can be TRUE or FALSE. <name> = Identifier for flag, up to 8 characters. <state> = Logic state, 1 or 0.	
GET	GET <param>	Instruction that retrieves the specified information from non-volatile memory (NVM). <param> = ALL: All vars, flags, and program space. <param> = VARS: Variables only. <param> = FLAGS: Flags only. <param> = PGM: Program space. <param> = IOS: I/O settings.	If <param> is not specified then <param> = ALL
INC	INC <var>	Instruction increments the specified variable by 1.	
INPUT	INPUT <var>, <param>	Instruction used to request input from the user. <param> is an optional nowait parameter. <var> = Variable. <param> = 0: Suspend prog. execution. <param> = 1: Do not suspend prog. execution.	If <param> is not specified then <param> = 0
INPUT1	INPUT1 <var>, <param>	Enhancement to the INPUT instruction which will only accept input from LYNX/MicroLYNX COMM 1. <param> is an optional nowait parameter. <var> = Variable. <param> = 0: Suspend prog. execution. <param> = 1: Do not suspend prog. execution.	If <param> is not specified then <param> = 0
INPUT2	INPUT2 <var>, <param>	Enhancement to the INPUT instruction which will only accept input from LYNX/MicroLYNX COMM 2. <param> is an optional nowait parameter. <var> = Variable. <param> = 0: Suspend prog. execution. <param> = 1: Do not suspend prog. execution.	If <param> is not specified then <param> = 0
IP	IP <param>	Initializes specified parameters to the factory default state. <param> = ALL: All vars, flags and I/O settings. <param> = VARS: Variables only. <param> = FLAGS: Flags only. <param> = IOS: I/O settings.	If <param> is not specified then <param> = ALL
PGM	GET PGM	Keyword used with the GET instruction to retrieve the contents of program space from NVM.	
PGM	PGM	Instruction used to place the LYNX/MicroLYNX in program mode.	
PRINT	PRINT <text/param>	Instruction used to output text and parameter value(s) to the host PC. See Language Reference for usage details.	
PRINT1	PRINT1 <text/param>	Enhancement to the PRINT instruction that will allow the user to only output to LYNX/MicroLYNX COMM 1.	
PRINT2	PRINT2 <text/param>	Enhancement to the PRINT instruction that will allow the user to only output to LYNX/MicroLYNX COMM 2.	
SAVE	SAVE	Saves all variables, flags and programs currently in working memory to NVM.	
SER	PRINT SER	Read-only variable that contains the serial number of the LYNX product.	
SET	SET <var>=<num> SET <flg>=<state>	Instruction used to set a variable or a flag to a specified value. NOTE: The SET instruction does not need to be entered to take effect. When entering <var>=<num> or <flg>=<state> SET is assumed.	

Data Related Instructions, Keywords and Flags cont'd			
Command	**Usage Example**	**Description**	**Default**
STATS	PRINT STATS	Keyword used with the PRINT instruction to output the values of only status flags.	
UFLGS	PRINT UFLGS	Keyword used with the PRINT instruction to output the values of only user defined flags.	
ULBLS	PRINT ULBLS	Keyword used with the PRINT instruction to output the values of only user defined labels.	
UVARS	PRINT UVARS	Keyword used with the PRINT instruction to output the values of only user defined variables.	
VAR	VAR = <name>	Instruction used to define a user variable to hold numeric data. <name> = 1 to 8 alpha-numeric characters.	
VARS	PRINT VARS GET VARS IP VARS	Keyword used with the GET, PRINT and IP instructions to indicate the inclusion of only variables.	
VER	PRINT VER	Read-only variable that contains the version information of the LYNX product.	

Event (Trip)

Event (Trip) Related Instructions, Keywords and Flags cont'd			
Command	**Usage Example**	**Description**	**Default**
TI1, TI2, TI3, TI4	TI<x>=<input>,<lbl/addr>,<output>	Trip on input variables which setup an input event (trip) for the specified input. This variable was formerly IT<x>. <x> = 1 - 4 <input> = Input used for trip. <lbl/addr> = Subroutine label or address to be activated on trip. <output> = I/O # to be set true on input trip.	0, 0, 0
TIE1, TIE2, TIE3, TIE4	TIE<x>=<flg>	Flag enables/disables the corresponding event trip. Flag was formely ITE<x>. <x> = 1 - 4 <flg> = 0: Disabled <flg> = 1: Enabled	0
TP1, TP2, TP3, TP4	TP<x>=<pos>,<lbl/addr>,<output>	Trip on position variables which setup an input event (trip) for the specified input. <x> = 1 - 4 <pos> = Position used for trip. <lbl/addr> = Subroutine label or address to be activated on trip. <output> = I/O # to be set true on input trip.	0.000, 0, 0
TPE1, TPE2, TPE3, TPE4	TPE<x>=<flg>	Flag enables/disables the corresponding event trip. <x> = 1 - 4 <flg> = 0: Disabled <flg> = 1: Enabled	0

Event (Trip) Related Instructions, Keywords and Flags cont'd			
Command	**Usage Example**	**Description**	**Default**
TT1, TT2, TT3, TT4	TTx=<time>,<lbl/addr>,<output>	Timer trip variables which setup an input event (trip) for the specified input. This variable was formerly TI<x>. <x> = 1 - 4 <time> = Time used for trip. <lbl/addr> = Subroutine label or address to be activated on trip. <output> = I/O # to be set true on input trip.	0, 0, 0
TTE1, TTE2, TTE3, TTE4	TTE<x>=<flg>	Trip on timer enable, enables the corresponding timer event trip (TT). Formerly TIE<x>. <x> = 1 - 4 <flg> = 0: Disabled <flg> = 1: Enabled	0
TTR1, TTR2, TTR3, TTR4	TTR<x>=<flg>	Trip on timer repeat, specifies whether or not the corresponding timer event trip (TT) will repeat each time the specified period expires. Formerly TIR<x> <x> = 1 - 4 <flg> = 0: Do not repeat. <flg> = 1: Repeat.	0
TV	TV=<velocity>,<lbl/addr>,<output>	Trip on velocity. This variable was formerly VT. <velocity> = Velocity used for trip. <lbl/addr> = Subroutine label or address to be activated on trip. <output> = I/O # to be set true on velocity trip.	0.000, 0, 0
TVE	TVE=<flg>	Trip on velocity enable. Enables the corresponding velocity trip. Formerly VTE. <flg> = 0: Disabled <flg> = 1: Enabled	0

Instructions That Can Be Used in a LYNX Program

Instructions That Can Be Contained in a LYNX Program			
Command	**Usage Example**	**Description**	**Default**
BR	BR <lbl/addr>, <param>	Performs conditional or unconditional branch to a routine in a LYNX program. <lbl/addr> = Subroutine label or address. <param> = Condition parameter: sets the condition for the branch. If blank, branch will be unconditional.	
CALL	CALL <lbl/addr>, <param>	Allows the user to invoke a subroutine within a LYNX program. <lbl/addr> = Subroutine label or address to be invoked if <param> = TRUE. <param> = Condition parameter: Flags or logical functions to be evaluated.	
CPL	CPL <var/flg>	Instruction that performs the two's complement of the specified variable or flag. <var/flg> = Variable or flag.	
DEC	DEC <var>	Instruction used to decrement the specified variable by 1. <var> = Variable.	
DELAY	DELAY=<time>	Delay program execution for specified <time>. <time> = Time in milliseconds.	

Instructions That Can Be Contained in a LYNX Program cont'd			
Command	**Usage Example**	**Description**	**Default**
DVF	DVF <param1>,<param2>	Instruction that deletes user defined variables and flags. <param1> = 0: All user vars and flags deleted. <param1> = 1: Only user vars deleted. <param1> = 2: Only user flags deleted. <param2> = 0: All global and local user vars and/or flags deleted. <param2> = 1: Only global user vars and/or flags deleted. <param2> = 2: Only local user vars and/or flags deleted.	If no parameter is specified, both will be 0
END	END	Stops the execution of a LYNX program.	
FIOS	FIOS<num1>,<num2>,<line>	Find I/O switch instruction. Parameters are optional. <num1> = ± speed in user units/sec. <num2> = ± creep in user units/sec. <line> = I/O line.	If not specified: <num> = VM <param> = VI
FLG	FLG <name>=<state>	Instruction to define a user flag that can be TRUE or FALSE. <name> = Identifier for flag, up to 8 characters <state> = Logic state, 1 or 0	
GET	GET <param>	Instruction that retrieves the specified information from non-volatile memory (NVM). <param> = ALL: All vars, flags and program space. <param> = VARS: Variables only. <param> = FLAGS: Flags only. <param> = PGM: Program space. <param> = IOS: I/O settings.	If <param> is not specified then <param> = ALL
HOLD	HOLD <mode>	Hold program execution until the specified motion phase completes. <mode> = 0: Program suspended until position change completes. <mode> = 1: Program suspended until velocity change completes. <mode> = 2: Program suspended until motion ceases.	0
INC	INC <var>	Instruction increments the specified variable by 1.	
INPUT	INPUT <var>, <param>	Instruction used to request input from the user. <param> is an optional nowait parameter. <var> = Variable. <param> = 0: Suspend prog. execution. <param> = 1: Do not suspend prog. execution.	If <param> is not specified then <param> = 0
INPUT1	INPUT1 <var>, <param>	Enhancement to the INPUT instruction which will only accept input from LYNX/MicroLYNX COMM 1. <param> is an optional nowait parameter. <var> = Variable. <param> = 0: Suspend prog. execution. <param> = 1: Do not suspend prog. execution.	If <param> is not specified then <param> = 0
INPUT2	INPUT2 <var>, <param>	Enhancement to the INPUT instruction which will only accept input from LYNX/MicroLYNX COMM 2. <param> is an optional nowait parameter. <var> = Variable. <param> = 0: Suspend prog. execution. <param> = 1: Do not suspend prog. execution.	If <param> is not specified then <param> = 0
IP	IP <param>	Initializes specified parameters to the factory default state. <param> = ALL: All vars, flags and I/O settings. <param> = VARS: Variables only. <param> = FLAGS: Flags only. <param> = IOS: I/O settings.	If <param> is not specified then <param> = ALL

Instructions That Can Be Contained in a LYNX Program cont'd			
Command	**Usage Example**	**Description**	**Default**
LBL	LBL=<name>	This instruction will label the address of a program or a subroutine within a program. <name> = 1 to 8 alphanumeric charaters including "_" underscore.	
MOVA	MOVA <num>, <mode>	Perform point-to-point move or index to an absolute position instruction. Use of <mode> is optional. <num> = Absolute position. <mode> = 0: Motion ceases when position is reached. <mode> = 1: Motion part of a profile, does not decelerate.	Mode 0 is used when mode not specified
MOVR	MOVR <num>, <mode>	Perform point-to-point move or index to a relative position instruction. <num> = Absolute position. <mode> = 0: Motion ceases when position is reached. <mode> = 1: Motion part of a profile, does not decelerate.	Mode 0 is used when mode not specified
NOP	NOP	No operation instruction, used to fill one byte of program space.	
ONER	ONER <lbl/addr>	On error, go to the specified label or address <lbl/addr>.	
PRINT	PRINT <text/param>	Instruction used to output text and parameter value(s) to the host PC. See Language Reference for usage details.	
PRINT1	PRINT1 <text/param>	Enhancement to the PRINT instruction that will allow the user to only output to LYNX/MicroLYNX COMM 1.	
PRINT2	PRINT2 <text/param>	Enhancement to the PRINT instruction that will allow the user to only output to LYNX/MicroLYNX COMM 2.	
RET	RET	Return statement RET is required at the end of a subroutine invoked by a CALL instruction.	
RUN	RUN <lbl/addr>	The RUN instruction executes a background taskto be run at the specified label or address <lbl/addr>.	
SAVE	SAVE	Saves all variables, flags and programs currently in working memory to NVM.	
SET	SET <var>=<num> SET <flg>=<state>	Instruction used to set a variable or a flag to a specified value. NOTE: The SET instruction does not need to be entered to take effect. When entering <var>=<num> or <flg>=<state> SET is assumed.	
SLEW	SLEW <num>=<mode>	Slew the motor at a constant velocity instruction. <vel> = User units/sec. <mode> = 0: Use acceleration ramp. <mode> = 1: Do not use acceleration ramp.	Mode 0 used if <mode> not specified
SSTP	SSTP<mode>	Stop the current motion using the specified deceleration profile and optionally stop the program. <mode> = 0: Stop motion only. <mode> = 1: Stop motion and program.	Mode 0 used if <mode> not specified
VARS	PRINT VARS GET VARS IP VARS	Keyword used with the GET, PRINT and IP instructions to indicate the inclusion of only variables.	

Instructions That Can Be Issued in Immediate Mode			
Command	**Usage Example**	**Description**	**Default**
CP	CP <lbl/address>, <flg>	Clear program instruction clears the program space in working memory beginning with the label or address specified by <lbl/addr>. <flg> = 0: Clear specified program only. <flg> = 1: Clear entire program space beginning with secified <lbl/addr>.	If <flg> not specified <flg>=0
CPL	CPL <var/flg>	Instruction that performs the two's complement of the specified variable or flag. <var/flg> = Variable or flag.	
DEC	DEC <var>	Instruction used to decrement the specified variable by 1. <var> = Variable.	
DVF	DVF <param1>,<param2>	Instruction that deletes user defined variables and flags. <param1> = 0: All user vars and flags deleted. <param1> = 1: Only user vars deleted. <param1> = 2: Only user flags deleted. <param2> = 0: All global and local user vars and/or flags deleted. <param2> = 1: Only global user vars and/or flags deleted. <param2> = 2: Only local user vars and/or flags deleted.	If no parameter is specified, both will be 0
END	END	Stops the execution of a LYNX program.	
EXEC	EXEC <lbl/addr>, <mode>	Execute the program label or located at address specified by <lbl/addr>. <mode> = 0: Normal execution. <mode> = 1: Trace mode. <mode> = 2: Single step mode.	If <mode> not specified <mode>=0
FIOS	FIOS<num1>,<num2>,<line>	Find I/O switch instruction. Parameters are optional. <num1> = ± Speed in user units/sec. <num2> = ± Creep in user units/sec. <line> = I/O line.	If not specified: <num> = VM <param> = VI
FLG	FLG <name>=<state>	Instruction to define a user flag that can be TRUE or FALSE. <name> = Identifier for flag, up to 8 characters. <state> = Logic state, 1 or 0.	
GET	GET <param>	Instruction that retrieves the specified information from non-volatile memory (NVM). <param> = ALL: All vars, flags and program space. <param> = VARS: Variables only. <param> = FLAGS: Flags only. <param> = PGM: Program space. <param> = IOS: I/O settings.	If <param> is not specified then <param> = ALL
INC	INC <var>	Instruction increments the specified variable by 1.	
IP	IP <param>	Initializes specified parameters to the factory default state. <param> = ALL: All vars, flags, and I/O settings. <param> = VARS: Variables only. <param> = FLAGS: Flags only. <param> = IOS: I/O settings.	If <param> is not specified then <param> = ALL

Instructions That Can Be Issued in Immediate Mode cont'd			
Command	**Usage Example**	**Description**	**Default**
LIST	LIST <lbl/addr>, <flg>	List stored program space beginning with label or address specified by <lbl/addr>. <flg> = 0: List through end of program space. <flg> = 1: List through first END.	If <lbl/addr> not specified will list all program space, If <flg> not specified <flg> = 0
MOVA	MOVA <num>, <mode>	Perform point-to-point move or index to an absolute position instruction. Use of <mode> is optional. <num> = Absolute position. <mode> = 0: Motion ceases when position is reached. <mode> = 1: Motion part of a profile, does not decelerate.	Mode 0 is used when mode not specified.
MOVR	MOVR <num>, <mode>	Perform point-to-point move or index to a relative position instruction. <num> = Absolute position. <mode> = 0: Motion ceases when position is reached. <mode> = 1: Motion part of a profile, does not decelerate.	Mode 0 is used when mode not specified
ONER	ONER <lbl/addr>	On error, go to the specified label or address <lbl/addr>.	
PAUS	PAUS	Suspends the executing program as well as any motion that is in progress.	
RES	RES	Resumes program and motion suspended by the PAUS instruction.	
SAVE	SAVE	Saves all variables, flags and programs currently in working memory to NVM.	
SET	SET <var>=<num> SET <flg>=<state>	Instruction used to set a variable or a flag to a specified value. NOTE: The SET instruction does not need to be entered to take effect. When entering <var>=<num> or <flg>=<state> SET is assumed.	
SLEW	SLEW <num>=<mode>	Slew the motor at a constant velocity instruction. <vel> = User units/sec. <mode> = 0: Use acceleration ramp. <mode> = 1: Do not use acceleration ramp.	Mode 0 used if <mode> not specified.
SSTP	SSTP<mode>	Stop the current motion using the specified deceleration profile and optionally stop the program. <mode> = 0: Stop motion only. <mode> = 1: Stop motion and program.	Mode 0 used if <mode> not specified
VARS	PRINT VARS GET VARS IP VARS	Keyword used with the GET, PRINT and IP instructions to indicate the inclusion of only variables.	
<esc>	Escape Key	Terminates all active operations and all running programs.	
^C	CTRL+C Keys	Terminates all active operations and all running programs, forces a partial reset of the LYNX or MicroLYNX.	

Miscellaneous and Setup Variables

Miscellaneous and Setup Variables			
Command	**Usage Example**	**Description**	**Default**
BAUD	BAUD = <param>	Sets the BAUD rate for serial communications with the LYNX/MicroLYNX. <param> = 48: 4800 bps <param> = 96: 9600 bps <param> = 19: 19,200 bps <param> = 38: 38,000 bps	9600 bps
BKGDA	BKGDA = <num>	Variable holds the present instruction address for the background task. <num> = 7 - 8175	
BREAK	BREAK =<num>, <lbl/addr>	Variable allows user to set up to 10 break points within a LYNX program. <num> = 0: Function disabled. <num> = 1-10: Program addresses specified by <lbl/addr> where execution will break.	
DISP	DISP = <lines>, <chars>, <wrap>	Specifies the display format for the PRINT instruction. <lines> = Number of lines (0-255, 0=no limit). <chars> = Number of characters (0-255, 0=no limit). <wrap> = 0: Do not wrap line. <wrap> = 1: Wrap long lines to next line.	0, 0, 0
DN	DN = <char>	Variable stores the device name to be used when in PARTY mode of operation. <char> = A - Z, a - z, 0 - 9	!
ECHO	ECHO = <mode>	Specifies whether or not the LYNX or MicroLYNX will echo commands received via communications port back over the line. <mode> = 0: Full duplex. <mode> = 1: Half duplex. <mode> = 2: Only respond to PRINT and LIST commands.	0
PAUSM	PAUSM=<mode>	Determines how motion is stopped in response to the PAUS instruction and whether or not it is restarted in response to the RES instruction. <mode> = 0: Normal DECL, resume with RES. <mode> = 1: LDECL deceleration, resume with RES. <mode> = 2: Complete motion, normal DECL. <mode> = 3: Complete motion, LDECL deceleration. <mode> = 4: Normal DECL, no resume. <mode> = 5: LDECL deceleration, no resume	0
PFMT	PFMT=<num1>, <num2>, <param>	Specifies the print format for numeric values. <num1> = # of digits before decimal (0-16). <num2> = # of digits after decimal (0-16). <param> = 0: Spaces as placeholders. <param> = 1: Zeros as placeholders. <param> = 2: No padding.	10, 3, 2
PRMPT	PRMPT=<char>	Specifies the character to be used by the LYNX or MicroLYNX as a prompt character. <char> = Character or ASCII decimal value (32-254).	> (ASCII 62)

Miscellaneous and Setup Flags

Miscellaneous and Setup Flags			
Command	**Usage Example**	**Description**	**Default**
BIO	BIO=<flg>	Sets communication mode. <flg> = 0: ASCII. <flg> = 1: Binary.	0
BKGD	BR<lbl/addr>, BKGD=<flg> PRINT BKGD	Read only status flag indicates whether or not a background program is running. <flg> = 0: Background program not running. <flg> = 1: Background program running.	0
BSY	PRINT BSY	Read only status flag indicates whether or not a program is running. Response = 0: Program not running. Response = 1: Program running.	0
CSE	CSE=<flg>	Enables/disables use of checksum when binary communications are used. <flg> = 0: Disabled. <flg> = 1: Enabled.	0
GECHE	GECHE=<flg>	Enables/disables the echo of global commands for use in party mode. <flg> = 0: Disabled. <flg> = 1: Enabled.	0
HELD	PRINT HELD	Read only status flag indicates whether or not a program is waiting for a position change, velocity change or motion to complete. Response = 0: Program not held. Response = 1: Program held.	0
HOST	HOST=<flg>	Enables/disables the status of a LYNX or MicroLYNX as the host module in a multi-drop system. <flg> = 0: Disabled (not host). <flg> = 1: Enabled (host).	0
LOGO	LOGO =<flg>	Enables/disables the sign-on banner. <flg> = 0: Disabled. <flg> = 1: Enabled.	1
PARTY	PARTY=<flg>	Enables/disables party mode. <flg> = 0: Disabled. <flg> = 1: Enabled.	0
PAUSD	BR <lbl/addr>, PAUSD = <flg> PRINT PAUSD	Read only status flag indicates whether or not a program has been paused. <flg> = 0: Program not paused. <flg> = 1: Program paused.	0
QUED	QUED=<flg>	Enables/disables the ability of LYNX/MicroLYNX modules in a multi-drop system to receive broadcast commands. <flg> = 0: Disabled. <flg> = 1: Enabled.	0
STK	BR <lbl/addr>, STK = <flg> PRINT STK	Read only status flag indicates a program subroutine stack fault. <flg> = 0: No fault. <flg> = 1: Stack overflow or underflow.	

Mathematical and Logical Functions

All mathematical and logical functions are evaluated sequentially, there is no hierarchy of functions. Therefore, the equation 1 + 2 * 3 evaluates to 9, and not 7. All functions can be evaluated in immediate mode, although their real usefulness is in a control module program.

Mathematical and Logical Functions			
Function	**Symbol**	**Example**	**Binary Mode Opcode Hex (Decimal)**
Addition	+	VAR3 = VAR1 + VAR2	10h (16)
Subtraction	-	VAR3 = VAR1 - VAR2	11h (17)
Multiplication	*	VAR3 = VAR1 * VAR2	12h (18)
Division	/	VAR3 = VAR2 / VAR1	13h (19)
AND (bitwise)	&	VAR3 = VAR1 & VAR2 (bitwise) FLG3 = FLG1 & FLG2 (logical)	14h (20)
OR (bitwise)	|	VAR3 = VAR1 | VAR2 (bitwise) FLG3 = FLG1 | FLG2 (logical)	15h (21)
XOR (bitwise)	^	VAR3 = VAR1 ^ VAR2 (bitwise) FLG3 = FLG1 ^ FLG2 (logical)	16h (22)
NOT	!	FLG3 = !FLG1 (Usable for flags only).	17h (23)
Equal To	=	VAR = <num>, FLG = <0/1>, FLG1 = VAR1 = VAR2	18h (24)
Less Than	<	FLG1 =<0/1>, VAR1 < VAR2	19h (25)
Greater Than	>	FLG1 = <0/1>, VAR1 > VAR2	1Ah (26)
Less Than or Equal To	<=	FLG1 = <0/1> ,VAR1 <= VAR2	1Bh (27)
Greater Than or Equal To	>=	FLG1 = <0/1>, VAR1 >= VAR2	1Ch (28)
Not Equal	<>	FLG1 = <0/1>, VAR1<> VAR2	1Dh (29)
Sine	SIN	VAR1 = SIN VAR2	1Eh (30)
Cosine	COS	VAR1 = COS VAR2	1Fh (31)
Tangent	TAN	VAR1 = TAN VAR2	20h (32)
Arc Sine	ASIN	VAR1 = ASIN VAR2	21h (33)
Arc Cosine	ACOS	VAR1 = ACOS VAR2	22h (34)
Arc Tangent	ATAN	VAR1 = ATAN VAR2	23h (35)
Square Root	SQR	VAR1 = SQR VAR2	24h (36)
Absolute	ABS	VAR1 = ABS VAR2	25h (37)
Integer Part	INT	VAR1 = INT VAR2	26h (38)
Fractional Part	FRC	VAR1 = FRC VAR2	27h (39)

ACCL Variable	Peak Acceleration Variable			
Usage Example	**Unit**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
ACCL=<num>	User units per second^2	± .0000000000000001 to ± 9,999,999,999,999,999	1.000000.000	60h (96)

Description The ACCL Variable sets the peak acceleration that will be reached by the Control Module in user units per second^2, based upon the value of MUNIT. If the user units have not been set, then the value is in clock pulses per second^2. The actual acceleration profile is maintained by the ACLT variable. The value given by ACCL sets the maximum acceleration that the Control Module will reach.

Related Commands MUNIT, ACLT, ACL, PFMT

ACL Read Only Status Flag	Acceleration Flag		
Usage Example	**Function**	**Default**	**Binary Mode Opcode Hex (Decimal)**
BR <lbl/addr>, ACL BR <lbl/addr>, !ACL PRINT ACL	<flg> = FALSE (0): Not accelerating. <flg> = TRUE (1) Axis is accelerating.	FALSE (0)	B8h (96)

Description The ACL Flag is a read only flag. The flag will be in a logic TRUE or "1" state when the axis is accelerating. It will be logic FALSE "0" at all other times. It can be used to branch to a program subroutine for actions such as toggling an output while the axis is accelerating, for example: to power an LED indicator.

Related Commands ACCL

ACLT Variable	Acceleration Type Variable		
Usage Example	**Parameters**	**Default**	**Binary Mode Opcode Hex (Decimal)**
ACLT=<param>	<param>=0: User Defined. <param>=1: Linear. <param>=2: Triangle S-Curve. <param>=3: Parabolic. <param>=4: Sinusoidal S-Curve.	1 - Linear	61h (97)

Description

The ACLT Variable defines the type of curve that will be used to build acceleration. The acceleration profiles are defined as follows:

0 – User-defined acceleration profile. This will follow the user-defined points in the ACLTBL (acceleration table) for the acceleration profile.
1 – Constant (linear) acceleration.
2 – Triangle S-Curve profile.
3 – Parabolic profile.
4 – Sinusoidal S-Curve profile.

Comparison of Acceleration Types:

1 – Constant smooth (linear) acceleration from initial to max velocity.
2 – Triangle S-Curve profile.
3 – The parabolic profile best utilizes the speed torque characteristics of a stepper motor since the highest acceleration takes place at low speeds. It will, however, be the profile that results in the maximum jerk, and is not recommended for applications requiring smooth starting and stopping. Such applications would include those that pull a material or move liquid.
4 – The Sinusoidal S-Curve profile is very similar to #2, the triangle S-Curve. The main difference is that it has less jerk when starting or stopping.

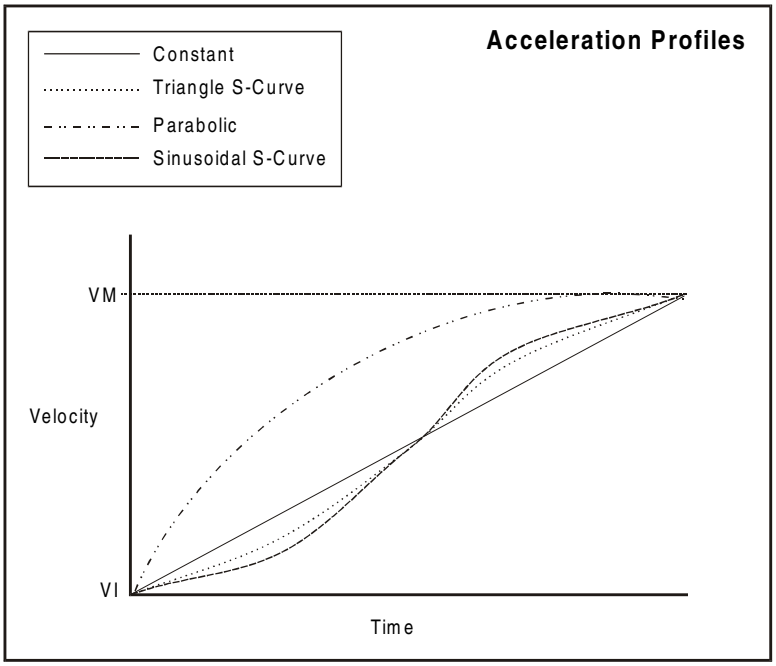


Figure 4.1: Acceleration Profiles

Related Commands

ACCL, ACLTBL

ACLTBL Variable	Acceleration Table Variable		
Usage Example	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
ACLT=<num>, <val>	<num> = 0 - 256 <val> = 0.00 - 1.00	Empty	62h (98)

Description

The acceleration table is a table of 256 points that can be used to define a user acceleration profile. The value specified in num 0 is the scale factor for the table. It will be multiplied by the rest of the points in the table to get the true acceleration profile. A point in the table can be specified by setting ACLTBL = num, val as shown in the example below. To use this, all 256 points must be defined.

If ACLTBL num 0 is set to 0 then the table is considered empty. In order for the table to be used, the ACLT, DCLT or LDCLT variable must be set to 0.

The routine below illustrates how the ACLT variable in all its types, effects the acceleration profile.

```
ACLTBL  =  1,0
ACLTBL  =  2,0.110
ACLTBL  =  3,0.220
```

```
ACLTBL  =  256,0.110
```

Related Commands

ACCL, ACLT, DCLT, LDCLT

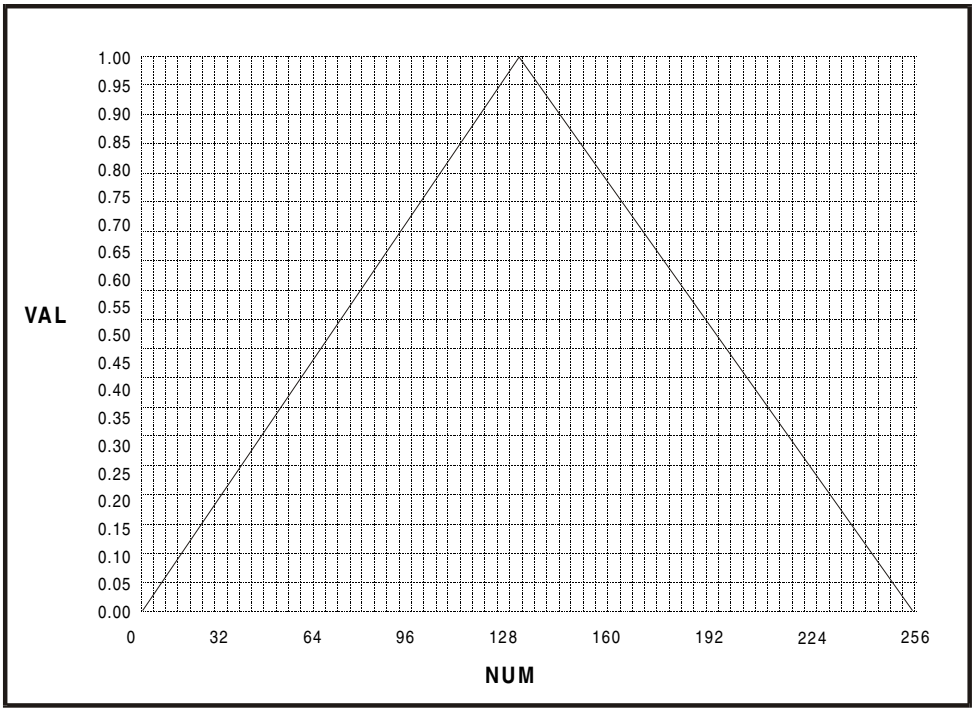


Figure 4.2: Triangle S-Curve Profile Plotted using <num>, <val> in an Acceleration Table

ADS Variable	Analog Input Setup Variable		
Usage Example	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
ADS<chan>=<aunit>,<func>, <law>	<chan> = 1 - 2 <aunit> = <func> = 1 - 2 <law> = 1 - 4	1, 1, 1	B3h (179)

Description

The ADS variable is used to set up the analog input functions of the analog input/joystick module. The following parameters are used:

<chan> = Channel # (1 or 2)
<aunit> = User Unit = MUNIT * AUNIT

<func> = 1: Analog input
<func> = 2: Joystick interface

<law> = 1: Linear
<law> = 2: Square law
<law> = 3: Cube law
<law> adjusts the joystick position to motor velocity transformation.

Related Commands

AIN, JSC, JSDB, JSFS, IJSC

ALL Keyword	Retrieve All Parameters
Usage Example	**Binary Mode Opcode Hex (Decimal)**
PRINT ALL IP ALL GET ALL	63h (99)

Description

The ALL keyword is used with GET, IP and PRINT instructions to signify that all types of parameters should be retrieved from nonvolatile memory (NVM), initialized to factory default values, or printed to the serial port.

When used with the GET instruction, all values of variables and flags are retrieved from NVM into working memory (RAM). In addition, the program space in working memory (RAM) is also refreshed from NVM. When used with the IP instruction, all system variables and flags in working memory (RAM) are restored to their factory default settings - user flags and variables are not affected. When used with the PRINT instruction, all variable and flag values are echoed to the host computer.

In order to save the changes made to working memory when ALL is used with the IP instruction, the SAVE instruction must be executed.

Related Commands

PRINT, IP, GET

AIN Read Only Variable	Read Analog Input Channel		
Usage Example	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
<var> = AIN <chan>	<var> = Variable to which data is saved. <chan> = 1 - 2		71h (113)

Description Read only variable causes a read of the analog input channel <chan>. Data is saved to the variable <var>.

Related Commands ADS, JSC, JSDB, JSFS, IJSC

BAUD Setup Variable	Baud Rate Variable			
Usage Example	**Unit**	**Parameters**	**Default**	**Binary Mode Opcode Hex (Decimal)**
BAUD=<param>	bps	<param> = 48: 4800 <param> = 96: 9600 <param> = 19: 19,200 <param> = 38: 38,000	9600 bps	64h (100)

Notes This variable sets the baud rate for serial communications with the control module. It sets the rate for both the RS-232 and RS-485 interfaces. The baud rate is set by indicating the first two digits of the desired rate as shown in the range section below.

In order for the new BAUD rate to take effect, the user must issue the SAVE instruction and then reset the Control Module. When the Control Module is reset, it will communicate at the new BAUD rate. NOTE: You will have to reset your terminal to the default setting of 9600 following any IP (Initialize Parameters) instruction to reestablish communications with the LYNX.

Related Commands SAVE

BIO Setup Flag	Binary Mode of Operation Flag		
Usage Example	**Function**	**Default**	**Binary Mode Opcode Hex (Decimal)**
BIO=<flg>	<flg> = FALSE (0):ASCII. <flg> = TRUE (1) Binary.	FALSE (0)	B9h 185)

Description This flag, when set to TRUE (1), sets the communications mode to binary. When the flag is FALSE (0), the communications mode is ASCII.

Related Commands CSE

BKGD Read Only Status Flag	Background Program Running		
Usage Example	**Status**	**Default**	**Binary Mode Opcode Hex (Decimal)**
BR<lbl/addr>, BKGD BR<lbl/addr>, !BKGD PRINT BKGD	BKGD = FALSE (0): Background program not running. BKGD = TRUE (1): Background Program running.	FALSE (0)	BAh (186)

Description This Read Only Status Flag indicates whether or not a background program is running. A background program is started by the RUN instruction. The result is two tasks: a foreground task and a background task running at the same time.

Related Commands RUN, BKGDA

BKGDA Read Only Status Variable	Background Program Address Variable		
Usage Example	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
PRINT BKGDA	1 - 8175		65h (101)

Description This variable holds the present instruction address for the background task.

Related Commands BKGD, RUN

BLE Setup Flag	Backlash Compensation Enable Flag		
Usage Example	**Function**	**Default**	**Binary Mode Opcode Hex (Decimal)**
BLE=<flg>	<flg> = FALSE (0): Disabled. <flg> = TRUE (1): Enabled.	FALSE (0)	BBh (187)

Notes Backlash could be described as the amount of mechanical variance in a system. For example, the nut on a leadscrew may not engage until several steps into the move. Again, during a direction change it would also take several steps for the actual motion in the opposite direction to commence. The LYNX Control Module is able to compensate for that amount using this feature with the BLM (Backlash Compensation Mode) and BLSH (Backlash Compensation Amount) Variables.

In order to use backlash compensation the function must be enabled. This flag will be used in conjunction with the BLM and BLSH variable to establish the type and amount of backlash compensation employed. The code example below illustrates how this flag might be used in setting up the backlash compensation for your system

Related Commands BLM, BLSH

BLM Setup Variable	Backlash Compensation Mode		
Usage Example	**Modes**	**Default**	**Binary Mode Opcode Hex (Decimal)**
BLM=<mode>	<mode> = 0: Mathematical compensation. <mode> = 1: Mechanical compensation.	Mode 0	66h (102)

Notes

Mode 0: Mathematical Compensation
When mathematical backlash is employed, a move made in the opposite direction of the previous move will have the value of BLSH added to it in the direction of the current move. This will have the effect of taking up the backlash resulting from the change in direction.

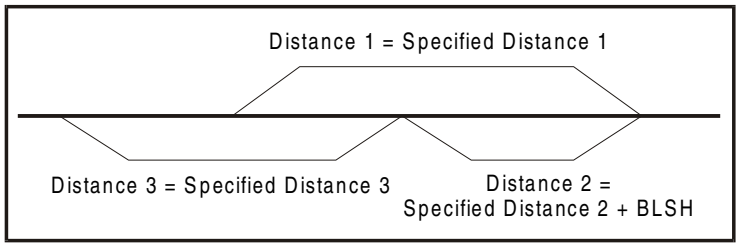


Figure 4.3: Mode 0 Backlash Compensation

Mode 1: Mechanical Compensation
When mechanical backlash compensation is employed, a move in the direction opposite to that indicated by the sign (±) of BLSH will have the value specified by BLSH added to it to take up the backlash. A separate move will be made to take up the backlash amount.

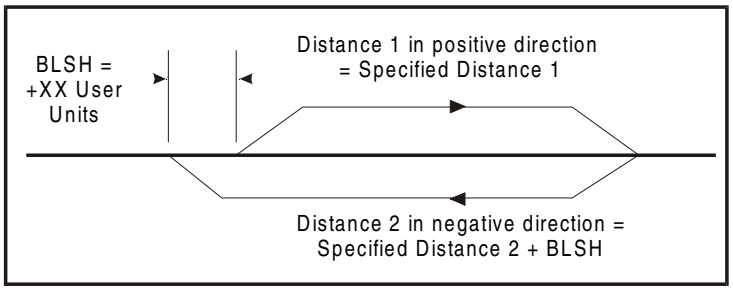


Figure 4.4: Mode 1 Backlash Compensation

Related Commands BLE, BLSH

BLSH Setup Variable	Backlash Compensation Amount Variable			
Usage Example	**Unit**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
BLSH=<num>	Userunits	±.0000000000000001 to ±9,999,999,999,999,999	0.000	67h (103)

Description This variable represents the amount of backlash compensation employed in user units (or clock pulses if MUNIT or EUNIT not specified) the sign indicates direction.

Related Commands BLE, BLM, MUNIT, EUNIT

BR Program Mode Instruction	Branch Instruction	
Usage Example	**Condition**	**Binary Mode Opcode Hex (Decimal)**
BR <lbl/addr>, <cond>	<lbl/addr> = Program or subroutine label or address. <cond> = Specified: Conditional branch. <cond> = Blank or unspecified: Unconditional branch.	30h (48)

Notes

The branch instruction can be used to perform a conditional or unconditional branch to a routine in a LYNX program. It can also be used to perform loops and IF THEN logic within a program.

There are two parameters to a branch instruction. These are used to perform two types of branches:

Conditional Branch
This type of branch first specifies a label or address where program execution should continue if the second parameter, the condition, is true. The condition parameter may include flags as well as logical functions that are to be evaluated.

Unconditional Branch
In this type of branch the second parameter is not specified, then the execution will continue at the address specified by the first parameter.

Syntax Examples

This section of code will use the branch instruction to execute a segment of code 10 times. In this case we will move a motor 10 user units 10 times. This usage is similar to a loop instruction in a higher level language.

```
VAR LOOPCNT = 0              'Create variable LOOPCNT, set to 0
PGM 100                      'Start program at address 100
LBL LOOPLBL                  'Label program address LOOPLBL
MOVR 10                      'Move the motor 10 user units
HOLD 2                       'Halt prog. execution until motion stops
DELAY 1000                   '1 second delay after motion stops
INC LOOPCNT                  'Increment the variable LOOPCNT
BR LOOPLBL, LOOPCNT<10       'Branch to LOOPLBL if LOOPCNT value is
                             'less than or equal to 10
PRINT "Done!"
END                          'End the program
PGM                          'Return to immediate mode
```

The following section of code will illustrate how a user could use the branch instruction to perform the equivalent of a DO-WHILE loop in a higher language. In this example, while the motor is accelerating, the velocity will be reported to the host terminal or terminal program running on a PC.

```
PGM 200                 'Start program at address 200
LBL CNTVEL              'Label address location CNTVEL
MUNIT = 51200/25        'Set the user units to Millimeters (arbitrary)
ACCL = 25               'Set acceleration to 25 mm/sec²
DECL = ACCL             'Set deceleration to 25 mm/sec²
VM = 200                'Set max velocity to 200 mm/sec
VI = VM/100             'Set initial velocity to 20 mm/sec
MOVR 2500               'Perform a relative move of 2500 mm
LBL DOWHILE             'Create subroutine DOWHILE
BR ENDWHILE,ACL = 0     'Conditional branch to routine ENDWHILE when the
                        'acceleration flag is equal to 0.
PRINT "Velocity = ",VEL, " millimeters per second"
BR DOWHILE              'Unconditional branch to routine DOWHILE
LBL ENDWHILE            'Create routine ENDWHILE
PRINT "Motor is at constant velocity =",VEL, " millimeters per second"
END                     'End the Program
PGM                     'Return to Immediate Mode
ENDENPGM PRINT "Done"
```

BREAK Variable	Break Point Variable		
Usage Example	**Function**	**Default**	**Binary Mode Opcode Hex (Decimal)**
BREAK=<num>, <lbl/addr>	<num> = 0: Function disabled. <num> = 1 - 10: Break points. <lbl/addr> = Program label or address where execution will break.	<num> = 0	68h (104)

Notes

Break allows the user to set break points within a LYNX program for help in debugging the program. When the program is executed while there are break points set, the program executes continuously until the address or label specified by the break point is encountered. The user can then step through the program by pressing the space bar to execute a single line. If the user wishes to continue execution to another break point or to the end of the program, this can be done by pressing the enter key.

There are 11 entries in the break point table. The first entry (break 0) enables or disables the function. If it is set to 0 the function is disabled, any nonzero value enables the function. The remaining ten entries (break 1 – break 10) hold program addresses at which execution should break awaiting a command to continue from the user. The program address may be entered numerically or by label.

BSY Read Only Status Flag	Busy Flag		
Usage Example	**Status**	**Default**	**Binary Mode Opcode Hex (Decimal)**
PRINT BSY	BSY = FALSE (0): No program running. BSY = TRUE (1): Program running.	FALSE (0)	BCh (188)

Notes

The BSY flag is a read only status flag which will read TRUE (1) when a program is executing. It will be in a FALSE (0) state at all other times.

By setting an output to I/O Type 21, the LYNX Product will activate that output whenever the BSY Flag is TRUE.

Related Commands PRINT, EXEC, IOS

CALL Program Mode Instruction	Call Subroutine Instruction	
Usage Example	**Condition**	**Binary Mode Opcode Hex (Decimal)**
CALL <lbl/addr>, <cond>	<lbl/addr> = Subroutine label or address to be invoked if <cond> = TRUE. <cond> = Flag or logical function.	31h (49)

Notes

This function can be used to invoke a subroutine within a program. This allows the user to segment code and call a subroutine from a number of places rather than repeating code within a program.

There are two parameters to the CALL instruction. The first specifies the label or program address of the

subroutine to be invoked if the second parameter, the condition, is true. If the second parameter is not specified, the subroutine specified by the first parameter is always invoked. The condition parameter can include flags as well as logical functions that are to be evaluated.

The subroutine should end with a RET instruction. The RET instruction will cause program execution to return to the line following the CALL instruction.

Syntax Example

```
IOS 20=0,1                    'Set IO group 20 to Input, HIGH TRUE
PGM 100                       'Start program at address 100
LBL MAINPGM                   'Label program "MAINPGM"
MOVR 51200                    'Index 51,200 msteps relative to current pos.
HOLD 2                        'Suspend program until motion stops
DELAY 500                     'Delay 500 milliseconds
CALL WAITIN21, IO 21=1        'Invoke subroutine "WAITIN21" when IO line 21=TRUE
BR MAINPGM                    'Loop back to "MAINPGM"
       LBL WAITIN21           'Declare program subroutine "WAITIN21"
       MOVR 51200*5           'Index relative 256,000 msteps
       HOLD 2                 'Suspend program until motion stops
       DELAY 500              'Delay 500 milliseconds
       BR WAITIN21, IO 21=1   'Loop to beginning of subroutine while IO 21 is TRUE
       RET                    'Return to main program
END                           'End program
PGM                           'Return to immediate mode
```

Related Commands RET

CP Immediate Mode Instruction	Clear Program Instruction	
Usage Example	**Modes**	**Binary Mode Opcode Hex (Decimal)**
CP <lbl/addr>, <mode>	<lbl/addr> = Subroutine label or address to be cleared. <mode> = 0: Clear only specified program. <mode> = 1: Clear to end of working memory.	32h (50)

Notes

This instruction will clear the program space in working memory (RAM) as specified by the instruction parameters.

There are two parameters to the CP instruction. The first specifies the label or program address of the location at which the clear command should begin. The second indicates whether only the specified program or subroutine (0) or the entire program space beginning with the specified address or label (1) should be cleared. If the second parameter is omitted or is specified as 0, the program space is cleared only until the first END or RET is reached. However, if it is specified as 1, the program space is cleared to the end of the program space.

Remember that this instruction operates on working memory (RAM). In order to remove the programs from the program space for the next power up, a SAVE instruction must be executed to save the contents of working memory in permanent memory (NVM).

Syntax Examples

```
CP 1,1          'This will clear all of working memory
CP TSTPRG,0     'This will clear the program labeled TSTPRG only.
CP 2000,1       'Clear from line 2000 to the end of working memory space
CP 2000,0       'Clear from line 2000 to the first END or RET
```

Related Commands SAVE

CPL Immediate/Program Instruction	Twos Complement Instruction	
Usage Example	**Parameter**	**Binary Mode Opcode Hex (Decimal)**
CPL <var/flag>	<var/flag> = Variable or flag.	33h (51)

Notes

This instruction will perform the twos complement of the specified variable or flag.

Has the effect of negating a numerical value. For instance, a variable named TESTVAR has a value of 2. CPL TESTVAR will cause the value of TESTVAR to equal -2. In the case of flags it will also be negated. For example a flag named TESTFLAG = TRUE (1), then CPL TESTFLAG will cause TESTFLAG to be FALSE (0)

Syntax Examples

```
VAR TESTVAR = 2        'Declare user variable "TESTVAR", set value to 2
PGM 100                'Start program at address 100
  LBL TEST             'Label the program "TEST"
  PRINT TESTVAR        'Print the value of TESTVAR
  CPL TESTVAR          'Twos complement TESTVAR
  PRINT TESTVAR        'Print the value of TESTVAR
  END                  'End the program
  PGM                  'Return to immediate mode
```

CSE Setup Flag	Check Sum Enable Flag		
Usage Example	**Function**	**Default**	**Binary Mode Opcode Hex (Decimal)**
CSE=<flg>	<flg> = FALSE (0): Disabled. <flg> = TRUE (1): Enabled.	FALSE (0)	BDh (189)

Notes

When this flag is enabled and binary mode communications is being used, each command sent to the LYNX requires a checksum to be included as the last byte of the command. The checksum is only used in binary mode and is the low 8 bits of the 16 bit sum of the address field, character count field, command field, data fields and separators included in the message. Refer to the section Modes of Operation for more information about the format of commands in binary and ASCII modes.

Related Commands

BIO

CTR1 Register Variable	Clock #1 Counter Variable			
Usage Examples	**Unit**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
<var> = CTR1 <math> CTR<x> <var> = CTR1 <math> <var> PRINT CTR1 CTR1 = <num> BR <lbl/addr>, CTR1=<num>	User Units	±2,147,000,000	0	69h (105)

Notes

This variable contains the raw count representation of the clock pulses sent to the motor drive. If there is no encoder in use (EE = 0), then this value scaled using MUNITS will match the value in the POS variable. If there is an encoder in use (EE = 1), this value scaled using MUNITS can be compared to the POS value to determine the position error for the axis (in this case POS is based on CTR2).

CTR1 is associated with Clock 1 (Step Clock/Direction-Defaulted to Differential I/O channels 11 and 12). Refer to the IOS variable for information on how these channels are set up by default and how they can be changed for your system.

Although the value of CTR1 can be set by the user, it is probably not necessary for the user to set this value directly. The value is automatically updated by the LYNX software when the POS value is set. The value of CTR1 is effected when POS is changed regardless of whether an encoder is being used in the system or not (EE = 0 or 1).

The example below will use the value of CTR1 to calculate the position error when working with the encoder functions enabled. Note that the position error is in raw counts and not user units in this case.

```
VAR POSERR           'Define variable POSERR
EE = 1               'Enable the encoder function
MOVR 100             'Perform a relative move of 100 counts
HOLD 2               'Suspend program execution until move completes
POSERR = CTR1-CTR2   'Calculate position Error
PRINT POSERR         'Display position error
```

Related Commands POS, CTR2, MUNIT, EE, IOS,

CTR2 Register Variable	Clock #2 Counter Variable			
Usage Examples	**Unit**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
<var> = CTR2 <math> CTR<x> <var> = CTR2 <math> <var> PRINT CTR2 CTR2 = <num> BR <lbl/addr>, CTR2=<num>	User Units	±2,147,000,000	0	6Ah (106)

Notes

This variable contains the raw counts representation of the clock edges received from the encoder if one is connected to the LYNX Product. If the encoder is in use (EE = 1), then this value scaled by EUNITS is given in POS and the encoder feedback is registered. If the encoder is not being used (EE = 0), the value of CTR2 can be used by the user to manually verify the position of the axis (in this case POS is based on CTR1).

CTR2 is associated with Clock 2 (Default Differential I/O channels 13 and 14). Refer to the IOS variable for information on how these channels are set up by default and how they can be changed for your system. It should be noted that the clock type could effect the clock rate here. For instance, if a quadrature clock type is chosen, the actual count will be four times the number of lines. A 1000 line encoder would produce 4000 counts per revolution of the motor.

If the encoder is in use by the LYNX Product (EE = 1), then the value of CTR2 probably need not be set directly as the value will be modified by a change to POS. If, however, the encoder is not in use (EE = 0) but an encoder is connected to the system, the user may directly modify the value of CTR2 in order to set the reference with respect to the motor.

Related Commands POS, CTR1, EUNIT, EE, IOS, HAS, HAE,

CTR3 Register Variable	Clock #3 Counter Variable			
Usage Examples	**Unit**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
PRINT CTR3 CTR3 = <num>	Clock Pulses (IO15 and 16)	±2,147,000,000	0	6Bh (107)

Notes

This variable contains the raw counts representation of the clock seen on Differential I/O channels 15 and 16. This channel will typically be used to drive a second stepper drive as an event type input for a second encoder, or as the master clock input for the half axis mode.

Again, refer to the IOS variable for information on how these channels are set up by default and how they can be changed for your system. It should be noted that the clock type could effect the clock rate here. For instance, if a quadrature clock type is chosen, the actual count will be four times the number of lines. A 1000 line encoder would produce 4000 counts per revolution of the motor.

Related Commands

IOS, RATIO, MUNIT, RATIOE, HAS, HAE,

DCL Read Only Status Flag	Deceleration Flag		
Usage Example	**Function**	**Default**	**Binary Mode Opcode Hex (Decimal)**
BR <lbl/addr>, DCL BR <lbl/addr>, !DCL PRINT DCL	<flg> = FALSE (0): Not decelerating. <flg> = TRUE (1) Axis is decelerating.	FALSE (0)	BEh (190)

Notes

The Deceleration Flag is a read only status flag which will be TRUE (1) when the Control Module is decelerating the Axis. It will be FALSE (0) at all other times.

Related Commands

DECL, DCLT

DCLT Setup Variable	Deceleration Type Variable		
Usage Example	**Parameters**	**Default**	**Binary Mode Opcode Hex (Decimal)**
DCLT=<param>	<param>=0: User Defined. <param>=1: Linear. <param>=2: Triangle S-Curve. <param>=3: Parabolic. <param>=4: Sinusoidal S-Curve.	1 - Linear	6Ch (108)

Notes

The DCLT Variable defines the type of curve that will be used to build deceleration.

Comparison of Deceleration Types:

1 – Constant smooth (linear) deceleration from initial to max. velocity.
2 – Triangle S-Curve profile.
3 – The Parabolic profile best utilizes the speed torque characteristics of a stepper motor since the highest acceleration takes place at low speeds. It will, however, be the profile that results in the maximum jerk, and is not recommended for applications requiring smooth starting and stopping. Such applications would include those that pull a material or move liquid.
4 – The Sinusoidal S-Curve profile is very similar to #3, the triangle S-Curve. The main difference is that it has less jerk when starting or stopping.

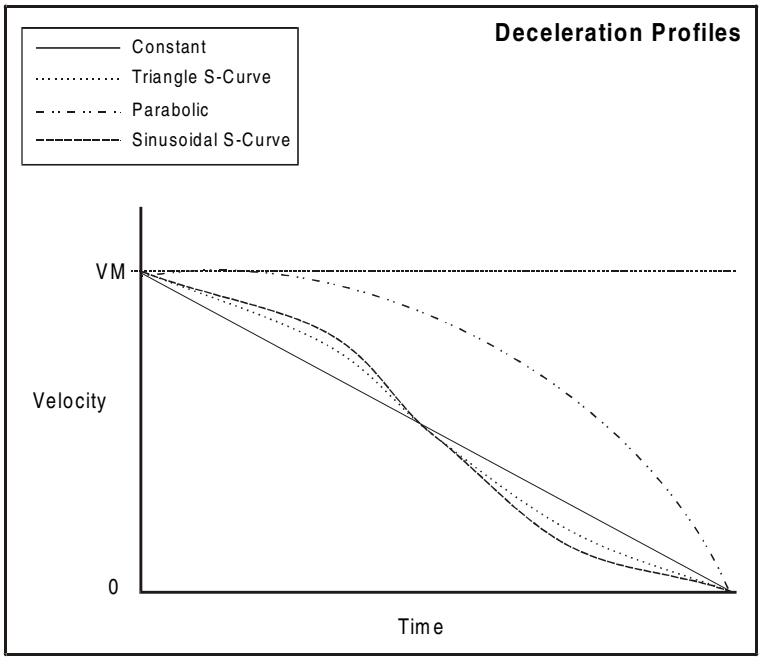


Figure 4.5: Deceleration Profiles

Related Commands DECL, ACLTBL

DECL Setup Variable	Peak Deceleration Variable			
Usage Example	**Unit**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
DECL=<num>	User Units per second^2	±.0000000000000001 to ±9,999,999,999,999,999	1.000000.000	6Dh (109)

Notes

The DECL Variable sets the peak deceleration that will be reached by the Control Module in user units per second^2. If the user units have not been set then the value is in clock pulses per second^2.

The actual deceleration profile is maintained by the DCLT variable. The value given by DECL sets the maximum deceleration that the Control Module will reach.

Related Commands MUNIT, DCLT, DCL

DEC Immediate/Program Instruction	Decrement Variable Instruction	
Usage Example	**Parameter**	**Binary Mode Opcode Hex (Decimal)**
DEC <var>	<var> = User or factory defined variable.	34h (52)

Description The Decrement Variable instruction will decrement the specified variable by one.

Syntax Example In the following example we will write a routine that will perform an operation in a loop 10 times.

```
VAR LOOPCTR =10              'Declare variable "LOOPCTR"
PGM 100                      'Start program at address 100
  LBL LOOP10                 'Label program "LOOP10"
  DEC LOOPCTR                'Decrement LOOPCTR variable
  PRINT "LOOPCTR=", LOOPCTR 'Print value of LOOPCTR variable
  HOLD 2                     'Suspend execution
  DELAY 1000                 'Delay 1 second
  BR LOOP10, LOOPCTR>0       'Loop to beginning of program while LOOPCTR >0
PRINT "DONE"
END
PGM
```

DELAY Program Mode Instruction	Delay Program Execution Instruction		
Usage Example	**Parameter**	**Range**	**Binary Mode Opcode Hex (Decimal)**
DELAY <time>	<time> = Time in milliseconds.	0 - 65535	35h (53)

Notes The Delay Instruction will delay program execution for a specified number of milliseconds before continuing.

The maximum delay time is 65,535 milliseconds or 65.535 seconds.

Syntax Example In the following example we will set an output, leave it set for 500 milliseconds and then clear it.

```
PGM 100              'Start program at address 100
LBL SAMPLE           'Label the program "SAMPLE"
IOS 21 = 0, 1        'Define I/O line 41 as a user defined output
IO 21 = 1            'Set I/O line 41 to TRUE (1)
DELAY 500            'Hold I/O line 41 in a TRUE (1) state for 500ms
IO 21 = 0            'Set I/O line 41 to FALSE (0)
END                  'End program
PGM                  'Return to immediate mode
```

DISP Setup Variable	Format Display Variable		
Usage Example	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
DISP=<lines>, <chars>, <wrap>	<lines> = 0 - 255 <chars> = 0 - 255 <wrap> = 0: Do not wrap lines. <wrap> = 1: Wrap lines.	0, 0, 0	6Eh (110)

Notes Specifies the display format for the print command. There are three parameters for this variable. The first, lines, gives the number of lines per screen. The second, chars, gives the number of characters per line. And the third, wrap, specifies whether or not to wrap long lines to the next line.

Related Commands PRINT

DN Setup Variable	Device Name Variable		
Usage Example	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
DN=<char>	ASCII Character: a - z, A - Z, 0 - 9	Exclamation Mark (!)	6Fh (111)

Notes The DN Variable stores the device name to be used when the LYNX Product is to be addressed in party mode operation.

The name is only used when party mode communications is being used (PARTY = 1). If the QUED flag is set, the LYNX Product will respond if addressed by its own name, or by the QUEUE or broadcast name "^".

All LYNX system nodes will respond if the name in a command is given as "*".

When the name is changed it must be saved into the nonvolatile memory if it is to be used in later sessions without being changed again.

Related Commands PARTY, QUED, SAVE

DRVEN Setup Flag	Drive Enable/Disable Flag		
Usage Example	**Function**	**Default**	**Binary Mode Opcode Hex (Decimal)**
DRVEN=<flg>	<flg> = FALSE (0): Disabled. <flg> = TRUE (1): Enabled.	TRUE (1)	BFh (191)

Notes The DRVEN flag enables or disables the drive module attached to the LYNX or MicroLYNX. This Flag is only relavent to drive modules, external drives are not affected by this flag.

Related Commands DRVTP, DRVRS

DRVRS Flag	Drive Reset Flag		
Usage Example	**Function**	**Default**	**Binary Mode Opcode Hex (Decimal)**
DRVRS=<flg>	<flg> = FALSE (0): Drive not reset. <flg> = TRUE (1): Reset drive.	FALSE (0)	C1h (195)

Notes The drive reset flag is a momentary flag which, when TRUE (1), will remain so for 10υs before returning to it's default (FALSE) state.

Related Commands DRVEN, DRVTP

DRVTP Read Only Variable	Drive Type Variable		
Usage Example	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
PRINT DRVTP	Response = 2: IM483 Response = 4: IM805		70h (112)

Notes The DRVTP variable provides a means to interogate the MicroLYNX to determine system configuration.

Related Commands DRVEN, DRVRS

DVF Immediate/Program Instruction	Delete User Defined Variables And Flags Instruction		
Usage Example	**Parameter**	**Default**	**Binary Mode Opcode Hex (Decimal)**
DVF <param1>, <param2>	<param1> = 0: All user variables and flags deleted. <param1> = 1: Only user variables deleted. <param1> = 2: Only user flags deleted. <param2> = 0: All global and local user vars and/or flags deleted. <param2> = 1: Only global user vars and/or flags deleted. <param2> = 2: Only local user vars and/or flags deleted.	0, 0	37h (55)

Notes This instruction deletes user-defined variables and flags.

Global variables and flags are defined in immediate mode, while local variables and flags are defined as part of a program.

Syntax Examples

```
DVF 1,2          'Delete only local variables
DVF 0,2          'Delete all local flags and variables
DVF 2,2          'Delete only local flags
DVF              'Delete all flags and variables
```

Related Commands VAR, FLG,

ECHO Setup Variable	Echo Mode Variable		
Usage Example	**Modes**	**Default**	**Binary Mode Opcode Hex (Decimal)**
ECHO=<mode>	<mode> = 0: Full duplex. <mode> = 1: Half duplex. <mode> = 2: No echo, only resp. to PRINT and LIST.	Full Duplex (1)	72h (114)

Notes

This variable specifies whether or not the Control Module should echo commands received via the communications port back over the line.

0 – Echo all information back over communications line (Full duplex).
1 – Don't echo the information, only send back prompt (Half duplex).
2 – Does not even send back prompt, only responds to PRINT and LIST commands.

EDB Setup Variable	Encoder Deadband Variable			
Usage Example	**Unit**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
EDB=<distance>	User Units	0 - 65535	2.000	73h (115)

Notes

This variable defines the + and - length of the encoder deadband for position maintenance.

When position maintenance is enabled, a move is made to the specified encoder position and when the move is complete, the LYNX Product maintains position within the specified deadband so that the position remains within (desired position – EDB < actual position < desired position + EDB).

The deadband position is specified in user units if EUNIT has been set. Otherwise, it is specified in encoder counts.

Related Commands

EUNIT, PME, EE

EE Setup Flag	Master Encoder Enable/Disable Flag		
Usage Example	**Function**	**Default**	**Binary Mode Opcode Hex (Decimal)**
EE=<flg>	<flg> = FALSE (0): Disabled. <flg> = TRUE (1): Enabled.	FALSE (0)	C5h (197)

Notes

This is the master enable for all of the encoder functions. It specifies whether or not position maintenance and/or stall detection should be performed if their individual enable flags are set.

If EE is TRUE but STLDE is FALSE, a stall will be detected but not acted upon. In other words, the STALL flag will become TRUE if the encoder does not keep up with the motor, but the motor will not be stopped as a result of the stall. Encoder feedback requires the use of I/O 13 and I/O 14 as the feedback input.

Related Commands

PME, STLDE, EDB, STALL

END Immediate/Program Instruction	END Program Instruction	
Usage Example	**Usage Rule**	**Binary Mode Opcode Hex (Decimal)**
END	Both immediate mode and program.	38h (56)

Notes Stops the execution of program. It should be the last line of a program written in memory.

If executed in immediate mode, the END instruction stops the execution of the current program as well as any background program that has been started by a RUN instruction.

Syntax Example A program will probably be identified by a label and run to the END statement. The following example program simply moves the motor to absolute 0.

```
PGM  100
LBL  ENDMOVE    'Label  Program  ENDMOVE
MOVA  0         'Perform  absolute  move  to  position  0
END             'End  program
PGM
```

Related Commands EXEC, RUN

ERR Read Only Status Flag	Error Flag		
Usage Example	**Status**	**Default**	**Binary Mode Opcode Hex (Decimal)**
BR <lbl/addr>, ERR BR <lbl/addr>, !ERR PRINT ERR	Response = FALSE (0): No error exists. Response = TRUE (1): Error exists.	FALSE (0)	C6h (198)

Notes The ERR flag is automatically cleared when a new program is executed. The only way to manually clear the ERR flag is to read the value of the ERROR variable.

By setting the type of an output to 23, the user can specify that the control module should activate the output whenever an error has occurred.

There is an instruction, ONER, which allows the user to specify the execution of a subroutine in the program memory when an error occurs. The subroutine might contain instructions to read the ERROR variable which would clear the ERR flag.

Related Commands ERROR, ONER, IOS

ERRA Read Only Variable	Error Address Variable		
Usage Example	**Response**	**Default**	**Binary Mode Opcode Hex (Decimal)**
PRINT ERRA Response = <addr1>, <addr2>	<addr1> = Foreground program address. <addr2> = Background program address.	0	B1h (177)

Notes The ERRA variable allows the user to troubleshoot programs and is automatically set when the ERROR flag is set. It contains the ERROR type and program location. It will clear only when it is replaced by another error address. ERRA will return two numbers. The first number will be the address of the last error in the foreground program, the second will be the address of the last error in the background program.

Related Commands ERROR, ONER, FAULT

ERROR Read Only Variable	Error Type Variable		
Usage Example	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
PRINT ERROR	See Error Table: Appendix B	0	74h (116)

Notes This read only variable indicates the program error code for the most recent error that has occurred in the Control Module. The ERROR variable must be read in order to clear the ERR flag.

See Appendix B for a list of possible errors.

Related Commands ERR, ONER, FAULT, ERRA

EUNIT Setup Variable	Encoder UnitsVariable			
Usage Example	**Unit**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
EUNIT=<num>	Encoder counts per user unit	±.0000000000000001 to ±9,999,999,999,999,999	1.000	75h (117)

Notes Conversion factor for converting motor steps or user units to encoder counts when an encoder is being used for position feedback.

When the encoder is enabled (EE = 1), POS will have the value of the scaled encoder counts. In other words, CTR2 / EUNIT will equal POS.

Note that if MUNIT is left at 1, then the user is programming in clock pulses and EUNIT should be the conversion of clock pulses to encoder counts. If the user wishes to program in a specified unit of measure such as millimeters, he/she must specify the units per clock pulse in MUNIT and units per encoder count in EUNIT.

Syntax Example In the example below the MUNIT and EUNIT variables will be set to measure position in degrees. In this example we will assume a stepper driver set ÷256 resolution with a 1.8° step motor with a 500 line quadrature encoder input. Since it is a quadrature input, we will multiply the encoder resolution by 4 to get the base EUNIT of 2000. To illustrate the use of the LYNX Control Module's math functions we will use the divide by (/) function. Allowing the LYNX to perform calculations will give us greater positional accuracy.

```
MUNIT = 51200/360     'Set MUNIT variable to use degrees as the user unit
EUNIT = 2000/360      'Set EUNIT variable to monitor position in degrees
```

NOTE: MUNIT AND EUNIT MUST BE DIVIDED BY THE SAME SCALING FACTOR!

Related Commands MUNIT, POS, EE

EXEC Immediate Mode Instruction	Execute Program Instruction	
Usage Example	**Modes**	**Binary Mode Opcode Hex (Decimal)**
EXEC <lbl/addr>, <mode>	<mode> = 0: Normal execution. <mode> = 1: Trace mode. <mode> = 2: Single step mode.	39h (57)

Notes

If the program to be executed is specified by a label, the EXEC instruction can be omitted. For instance, if a program is specified by the label TSTPRG, the command EXEC TSTPRG is equivalent to simply typing TSTPRG.

There are three modes of program execution.

Mode 0 Normal execution, is specified by a mode of 0 (or simply leaving the mode blank).

Mode 1 Trace mode is specified by a mode of 1. This means that the program executes continuously until the program END is encountered, but the instructions are "traced" to the communications port so the user can see what instructions have been executed.

Mode 2 Single step mode is specified by a mode of 2. In this mode, the user can step through the program using the space bar to execute the next line of the program. The program can be resumed at normal speed in this mode by pressing the enter key.

Syntax Examples

```
EXEC TSTPRG, 2        'Execute TSTPRG in single step mode.
EXEC 2000             'Execute program at line 2000 in normal mode.
```

Related Commands PAUS, END

FAULT Read Only Status Flag	Fault Indicator LED Enable/Disable Flag		
Usage Example	**Function**	**Default**	**Binary Mode Opcode Hex (Decimal)**
FAULT=<flg>	<flg> = FALSE (0): Disabled. <flg> = TRUE (1) Enabled.	TRUE (1)	EFh (239)

Notes

This Flag allows the user to enable or disable the red fault indication LED on the Control Module. When TRUE (1) will display all ERROR conditions by illuminating the Fault indicator LED. When FALSE (0) the Fault LED will not illuminate.

In order to clear the FAULT LED you must issue a PRINT ERROR statement.

Related Commands ERROR, ONER, IOS, ERRA

FIOS Immediate/Program Instruction	Find I/O Switch Instruction		
Usage Example	**Parameter**	**Default**	**Binary Mode Opcode Hex (Decimal)**
FIOS <speed>, <creep>, <line>	<speed> = ± speed in user units/sec. <creep> = ± creep in user units/sec. <line> = I/O line number.	<speed> = VM <creep> = VI	3Ah (58)

Notes

This instruction will find the selected I/O switch.

There are three optional parameters for this command:

1) Speed: Specifies the direction and speed that the axis will move until the switch is activated.

2) Creep: Specifies the direction and speed that the axis will move off the switch until it becomes inactive again.

3) Line: Specifies the Input switch to be monitored.

When FIOS is executed, the axis moves in the direction specified by the sign of speed at the speed until the input specified by line becomes active. It then creeps off of the switch in the direction specified by the sign of creep at the creep speed. Motion is stopped as soon as the switch becomes deactivated.

If speed is not specified, the speed used to find the switch is –VM. If creep is not specified, the speed used to move off of the switch is +VI. If line is not specified, the input specified as the home switch (IOS type 12) is monitored for activation.

If a limit switch is encountered before the specified switch is seen, the direction will be reversed until the specified switch is seen. The motor will be allowed to go by the switch and then return in the specified direction to find the home position.

If both limits are encountered before the specified switch is seen, the motion is stopped and an error is flagged.

Syntax Example

In this example we will use the FIOS command to home the axis on initial power up. We will not specify the line parameter since we want to use the home switch. We will specify the speeds, however. Assume that the MUNIT and EUNIT variables have been set so that the user unit is inches, therefore speeds are specified in inches per second. We will search for the switch at 5 inches per second and come off of it at .1 inch per second.

```
IOS 21=12            'Set IO line 21 to a homing input
PGM100               'Start program at address 100
LBL FINDIO           'Label program "FINDIO"
FIOS -5,+.1,21       'Find home switch at 5 in/sec, creep off at 0.1 in/sec
END                  'End program
PGM                  'Return to immediate mode
```

Related Commands

VM, VI, IOS

FLAGS Keyword	Retrieve Flags Keyword	
Usage Example		**Binary Mode Opcode Hex (Decimal)**
PRINT FLAGS IP FLAGS GET FLAGS		78h (120)

Notes Used with the GET, IP and PRINT commands to specify that all flags should be retrieved from nonvolatile memory (NVM), set to their factory default values, or printed to the serial port, respectively. When used with the GET instruction, only flag values are retrieved from NVM. When used with the IP instruction, only system flag values are set to the factory default parameters. In this case, user-defined flags are not affected. When used with the PRINT instruction, only flag values are echoed to the host computer.

Related Commands PRINT, GET, IP

FLG Immediate/Program Mode Instruction	Define User Flag Instruction	
Usage Example	**Parameters**	**Binary Mode Opcode Hex (Decimal)**
FLG <name>	<name> = 1 to 8 Alpha-numeric Characters +Underscore (_)	3Bh (59)

Notes The name of the flag can be 1 to 8 alphanumeric characters in length. You may use the underscore (_) character in the name as well. The value of the flag can be initialized when it is defined. If it is not specifically initialized, it will have a value of FALSE until it is set.

Flags can be “global” or “local”. A local flag is one that has been defined in a program while a global flag is defined in immediate mode. It should be noted that a local flag is not static, but is erased and declared again whenever the program is executed.

GET Immediate/Program Mode Instruction	Retrieve Variables and Flags Instruction	
Usage Example	**Parameters**	**Binary Mode Opcode Hex (Decimal)**
GET VARS GET FLAGS GET ALL		3Ch (60)

Notes Retrieves the specified information from nonvolatile memory (NVM) into working memory (RAM).

There is one optional parameter to this instruction. If there is no value given for this parameter or it is ALL, then all variables, flags and the program space are refreshed in working memory. Alternately, if the parameter is specified as FLAGS only the values of system flags are refreshed, and if the parameter is specified as VARS only the values of the system variables are refreshed.

It should be noted that user-defined flags and variables (those defined using a FLG or VAR instruction) are not refreshed with a GET command.

Related Commands ALL, FLAGS, VARS, PGM

GECHE Setup Flag	Global Echo Enable/Disable Flag		
Usage Example	**Function**	**Default**	**Binary Mode Opcode Hex (Decimal)**
GECHE=<flg>	<flg> = FALSE (0): Disabled. <flg> = TRUE (1) Enabled.	FALSE (0)	C8h (200)

Notes

Enable (1) or disable (0) the echo of Global commands. For use in party mode communications only.

A global command is any command that specifies the LYNX Product name as the GLOBAL Control module character "*" instead of a specific LYNX system node name.

This flag should be TRUE for only one LYNX node on the common RS-422 line.

Related Commands PARTY

HAE Setup Flag	Half Axis Mode Enable/Disable Flag		
Usage Example	**Function**	**Default**	**Binary Mode Opcode Hex (Decimal)**
HAE=<flg>	<flg> = FALSE (0): Disabled. <flg> = TRUE (1) Enabled.	FALSE (0)	C9h (201)

Notes

In half axis mode the master clock is taken from the clock input 2, 3 or 4 (line pairs 13-14, 15-16 or 17-18) which have been set for input, clock type and ratio enabled. The primary axis moves as a ratio of this clock based on the factor entered in HAS. This is an implementation of a master follower where the master is input into a clock input and the primary axis follows based on the specified factor.

Related Commands HAS

HAS Setup Variable	HAlf Axis Mode Scaling Variable			
Usage Example	**Unit**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
HAS=<param>	Scaling Factor	-1 <=Factor <1	1.000	79h (121)

Notes

In half axis mode the master clock is taken from a clock input 2, 3 or 4 (line pairs 13-14, 15-16 or 17-18) which have been set for input, clock type and ratio enabled. This is the factor at which the count rate out to the primary drive will follow the external clock in half axis mode. This is an implementation of a master follower where the master is input into the clock input and the primary axis follows based on the specified factor.

HAE must be set to TRUE in order to enable the function.

Related Commands HAE, IOS

HCDT Setup Variable	Hold Current Delay Time Variable			
Usage Example	**Unit**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
HCDT=<time>	milliseconds		0	7Ah (122)

Notes The HCDT variable sets the delay time in milliseconds between the cessation of motion and when the LYNX or MicroLYNX shifts to the holding current level specified by the MHC variable. The delay time is also effected by the MSDT (Motor Settling Delay Time) variable in that the total time from motion ceasing to current change is represented by the sum of MSDT + HCDT.

Related Commands MAC, MRC, MHC, MSDT

HELD Read Only Status Flag	Program Execution Held Flag		
Usage Example	**Status**	**Default**	**Binary Mode Opcode Hex (Decimal)**
BR <lbl/addr> HELD BR <lbl/addr> !HELD PRINT HELD	Status = FALSE (0): Program executing. Status = TRUE (1): Program suspended.	FALSE (0)	CAh (202)

Notes This flag is TRUE (1) when the program is waiting for the position change, velocity change or motion to complete.

Related Commands HOLD

HOLD Program Mode Instruction	Hold Program Execution During A Move Instruction	
Usage Example	**Modes**	**Binary Mode Opcode Hex (Decimal)**
HOLD <mode>	<mode> = 0: Suspend program until position change completes. <mode> = 1: Suspend program until velocity change completes. <mode> = 2: Suspend program until motion completes.	3Dh (61)

Notes Hold program execution until the specified motion phase completes. There is one optional parameter to the HOLD instruction which specifies how long the program execution should be held. If the parameter is 0 or not specified, the program will suspend until the position change completes (PCHG becomes FALSE). If the parameter is 1, the program will suspend until the velocity change completes (VCHG becomes FALSE). If the parameter is 2, the program will suspend until the motion completes (MVG becomes FALSE).

Syntax Example In this example we will start a motion and wait for the motion to complete before continuing with the program.

```
MOVR 10        'Perform a relative move of ten user units
HOLD 2         'Suspend program execution until motion completes
```

Related Commands HELD, PCHG, VCHG, MVG

HOST Setup Flag	Host Interface Enable/Disable Flag		
Usage Example	**Status**	**Default**	**Binary Mode Opcode Hex (Decimal)**
HOST=<flg>	<flg> = FALSE (0): Disabled. <flg> = TRUE (1): Enabled.	FALSE (0)	CBh (203)

Notes

This is the Host Interface flag. It is only relevant in a system that contains several LYNX Product nodes in a multi-drop configuration. When this flag is set, the node that will serve as the interface between the Host PC and the rest of the system is connected via the RS-232 port. Other LYNX Product nodes in the system are connected together via RS-485 interface.

To properly configure the system, the host computer should be connected to the Host Interface via RS-232. The remaining nodes in the system should then have their RS-485 RX inputs connected to the Host Interface Control module's RS-485 TX output, and their RS-422 TX outputs connected to the Host Interface's RS-485 RX input. The HOST flag of the Host Interface should be set. Host PC communications are received by the Host Interface Control module and forwarded to all of the other control modules in the system via the RS-485 channel. Responses from the Host Interface module are routed to the Host PC via the RS-232 channel, but are not seen by the other system nodes on the RS-485 channel. The Host Interface module to the Host PC via the RS-232 channel routes responses from the other control modules.

Only the Host Interface should have the HOST flag set. All other system nodes should have the flag cleared which allows the control modules to operate on commands received via either the RS-485 or RS-232 ports. In addition, the LYNX Products's responses are output to both ports.

It should be noted that there is a switch which allows the user to set the host flag in hardware, but the software overrides the hardware. Therefore, if switch is set for Host in hardware and the user sets the host flag to FALSE (0) in software, the unit will not act as a host interface.

Related Commands PARTY

IJSC Immediate/Program Instruction	Calibrate Joystick Instruction	
Usage Example	**Parameter**	**Binary Mode Opcode Hex (Decimal)**
IJSC		84h (132)

Notes

The IJSC instruction is a new addition to the LYNX instruction set. It is added to support the Analog Input/Joystick interface module when operating in joystick mode.

Execution of this command followed by moving the joystick over its range of motion and back to center, then pressing the "ENTER" key or allowing it to time out in 30 seconds will calibrate the joystick.

INC Immediate/Program Instruction	Increment Variable Instruction	
Usage Example	**Parameter**	**Binary Mode Opcode Hex (Decimal)**
INC <var>	<var> = Any user or factory defined variable.	3Fh (63)

Notes The Increment Variable instruction will increment the specified variable by one.

Syntax Example In the following example we will write a routine that will perform an operation in a loop 10 times.

```
VAR LOOPCTR = 0              'Declare variable LOOPCTR, set value to 0
PGM 100
LBL LOOP10                   'Declare subroutine LOOP10
INC LOOPCTR                  'Increment the value of LOOPCTR
PRINT "LOOPCTR=", LOOPCTR    'Display the value of LOOPCTR
DELAY 1000                   'Delay Program execution for 1 sec.
BR LOOP10, LOOPCTR<10        'Cond. branch to LOOP10 while LOOPCTR < 10
PRINT "DONE"
END
PGM
```

INPUT Program Mode Instruction	User Input Request Instruction	
Usage Example	**Parameter**	**Binary Mode Opcode Hex (Decimal)**
INPUT <var>, <param>	<var> = Any user or factory defined variable. <param> = 0: Suspend program execution while waiting for user input. <param> = 1: Do not suspend program execution.	40h (64)

Notes

Command to request input from the user over the RS-232 or RS-485 channel. The input must be numeric and is input into the variable that is specified as a parameter to the command.

This instruction has been modified since the prior release with the inclusion of the "no wait" parameter <param>. This parameter allows the user to determine whether or not the program execution will suspend while awaiting input from the user. If <param> = 0 or is not specified, program execution will suspend until the input request is satisfied. If <param> = 1, then program execution will continue uninterrupted.

It is up to the programmer to use the PRINT command to request the information from the user, before using the INPUT statement to accept the information into the specified variable. In order to keep the cursor on the same line as the user instructions, the string should be followed by a semicolon as shown in the following example.

The variable used as the parameter for the INPUT instruction may be a system or USER variable. If a USER variable is being used, it must be declared prior to the INPUT instruction using the VAR instruction.

Syntax Example In the following example we will write a routine that will request that the user input the velocity to be used for the next move

```
VAR SPEED                    'Declare "SPEED" variable
PGM 100                      'Start program at address 100
LBL SAMPLE                   'Label the program "SAMPLE"
PRINT "Input the velocity for the next move:";
INPUT SPEED                  'Input velocity
SLEW SPEED                   'Perform a relative move of ten user units
END
PGM
```

INPUT1 Program Mode Instruction	User Input Request Instruction (LYNX COMM1)	
Usage Example	**Parameter**	**Binary Mode Opcode Hex (Decimal)**
INPUT1 <var>, <param>	<var> = Any user or factory defined variable. <param> = 0: Suspend program execution while waiting for user input. <param> = 1: Do not suspend program execution.	57h (87)

Notes This is an enhancement of the INPUT instruction in that it will only accept input from LYNX COMM 1, otherwise it operates the same as the INPUT instruction.

INPUT2 Program Mode Instruction	User Input Request Instruction (LYNX COMM2)	
Usage Example	**Parameter**	**Binary Mode Opcode Hex (Decimal)**
INPUT1 <var>, <param>	<var> = Any user or factory defined variable. <param> = 0: Suspend program execution while waiting for user input. <param> = 1: Do not suspend program execution.	58h (88)

Notes This is an enhancement of the INPUT instruction in that it will only accept input from LYNX COMM 2, otherwise it operates the same as the INPUT instruction.

IO Variable	Read/Write IO Variable	
Usage Example	**Range**	**Binary Mode Opcode Hex (Decimal)**
PRINT IO <line/group> IO <line/group> = <0-1/0-63>	<line/group> = I/O lines (21-26, 31-36, 41-46, 51-56) or IO Group (20 - 50)	7Bh (123)

Notes

There are two types of I/O with the LYNX system. First, there can be up to eight (8) high speed differential I/O individually programmable as clock inputs or outputs or for general purpose use. If used as inputs, these are digitally filtered with a cutoff frequency that can be set by the user.

Second, there are up to twenty-four (24) general purpose I/O which can be used for special purpose inputs, such as limits or home, as well as general purpose inputs and outputs. As inputs, each is digitally filtered with a cutoff frequency that can be set by the user. For more details on I/O structure and availability by module see the section on Configuring the Digital IO, in the part of this document pertaining to the LYNX product purchased.

I/O is divided into the following groups.

Group 10 ..... Up to 8 High Speed Differential I/O line pairs.
Group 20 ..... General Purpose I/O lines 21 - 26
Group 30 ..... General Purpose I/O lines 31 - 36
Group 40 ..... General Purpose I/O lines 41 - 46
Group 50 ..... General Purpose I/O lines 51 - 56

Each digital I/O line can be programmed as Input or Output, as well as have its various functions such as triggering, High/Low TRUE, etc. using the IOS variable. The digital filtering for inputs can be set using the IOF variable.

You can report or change the state of individual inputs or outputs, or you can report or change the binary state of the entire group. In the former case, the response from the LYNX will be a 1 if the input or output is active, and a 0 if it is not. In the latter case, the response is a decimal equivalent of the byte that is a bitwise representation, or binary weight of the entire group.

If for some reason the I/O cannot be set (i.e. output shorted, held to True or 1) an error message will be generated. See: Appendix B: Error Table for more details.

Related Commands IOS, IOF

IOF Setup Variable	Digital Input Filtering Variable		
Usage Example	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
PRINT IOF <group> IOF <group> = <param>	<group> = 10 - 50 <param> = 1 - 7	IO Group 10 = 0 IO Groups 20 - 50 = 7	7Dh (125)

Notes

This variable sets the digital filtering to be applied to the specified I/O group.

When setting the digital filtering for the I/O, you must specify the group for which the filter should be applied. This can be group 1 (the high speed I/O) or groups 2 - 5 (the standard and optional I/O).

The filter values used for the high speed differential I/O are different than those used for the general purpose I/O.

IOF Settings for Differential IO (Group 10)

Filter Setting	Cutoff Frequency	Minimum Detectable Pulse Width
0 (default)	5.00 MHz	100 nanoseconds
1	2.50 MHz	200 nanoseconds
2	1.25 MHz	400 nanoseconds
3	625 kHz	800 nanoseconds
4	313 kHz	1.6 microseconds
5	156 kHz	3.2 microseconds
6	78.1 kHz	6.4 microseconds
7	39.1 kHz	12.8 microseconds

IOF Settings for General Purpose Isolated IO (Groups 20 - 50)

Filter Setting	Cutoff Frequency	Minimum Detectable Pulse Width
0	27.5 kHz	18 microseconds
1	13.7 kHz	36 microseconds
2	6.89 kHz	73 microseconds
3	3.44 kHz	145 microseconds
4	1.72 kHz	290 microseconds
5	860 Hz	581 microseconds
6	430 Hz	1.162 milliseconds
7 (default)	215 Hz	2.323 milliseconds

Related Commands IOS, IO

IOS Setup Variable	I/O Configuration Variable/Keyword		
Usage Example	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
See Below	See Below	See Below	7Eh (126)

Description Specifies the set up of the I/O. Is also used as a keyword for the IP instruction.

Usage IOS <Line/Group> = <type>, <i/o>, <h/l>, <l/e>, <clk type>, <ratio>

Default Settings

I/O Group 10

I/O	Function	IOS	Notes
11	CLK1A	1, 1, 1, 0, 2, 0	Step Clock Output
12	CLK1B	2, 1, 1, 0, 2, 0	Direction Output
13	CLK2A	3, 0, 1, 0, 1, 0	
14	CLK2B	4, 0, 1, 0, 1, 0	
15	CLK3A	5, 0, 1, 0, 1, 0	
16	CLK3B	6, 0, 1, 0, 1, 0	
17	CLK4A	7, 0, 1, 0, 1, 0	
18	CLK4B	8, 0, 1, 0, 1, 0	

I/O Groups 20 - 50

I/O	Function	IOS	Notes
21-26	USER	0, 0, 1, 0, 0, 0	
31-36	USER	0, 0, 1, 0, 0, 0	
41-46	USER	0, 0, 1, 0, 0, 0	
51-56	USER	0, 0, 1, 0, 0, 0	

Notes You can specify the set up for individual I/O or for the entire group of I/O. To specify the group, you would specify 10 for group 10, 20 for group 20, etc. Otherwise, simply specify the I/O number. There are six settings that can be specified for each I/O. The first setting is the I/O type <type>. The type can be one of the following:

Type	Function	Input/Output	Type	Function	Input/Output
0:	USER	Input or Output	13:	LIMIT PLUS	Input
1:	CLK1A	Output Only	14:	LIMIT MINUS	Input
2:	CLK1B	Output Only	15:	STATUS	Input
3:	CLK2A	Input or Output	16:	JOG PLUS	Input
4:	CLK2B	Input or Output	17:	JOG MINUS	Input
5:	CLK3A	Input or Output	18:	MVG	Output
6:	CLK3B	Input or Output	19:	PCHG	Output
7:	CLK4A	Input or Output	20:	VCHG	Output
8:	CLK4B	Input or Output	21:	BSY	Output
9:	GO	Input	22:	STALL	Output
10:	STOP	Input	23:	ERR	Output
11:	PAUSE	Input	24:	PAUSD	Output
12:	HOME	Input	25:	SYNC	Output

The second setting is Input or Output <i/o>:
0 = Input
1 = Output

The third setting is High/Low True <h/l>:
0 = Low True
1 = High True

The fourth setting is Level/Edge Triggering <l/e>:
0 = Level Triggered
1 = Edge Triggered

The fifth setting is Clock Type <clk type>:
(Differential I/O Only)
0 = No Clock
1 = Quadrature
2 = Step/Direction
3 = Up/Down

The sixth setting is Ratio <ratio>:
(Differential I/O Only)
0 = No Ratio
1 = Ratio Mode

Syntax Examples

```
IOS 20 = 0          'Set all the inputs in Group 20 to user defined.
IOS 21 = 10,0,1,1   'Set I/O Line 21 to a Stop Input, High True, Edge Triggered.
```

A more detailed discussion on configuring the digital I/O using the IOS variable can be found in I/O configuration section of the part of this document pertaining to the LYNX product purchased.

Related Commands IOF, IO, IP

IP Immediate/Program Instruction	Initialize Parameters Instruction

Usage Example	Binary Mode Opcode Hex (Decimal)
IP ALL IP VARS IP FLAGS IP IOS	41h (65)

Notes

Initializes specified parameters to the factory defaults in working memory (RAM).

To specify which kind of parameters should be initialized, use the following keywords:

ALL (or blank)	All variables, flags, and I/O settings (IOS)
VARS	Variables only
FLAGS	Flags only
IOS	I/O only

If you want the factory default settings to permanently replace the contents of the specified parameter type in NVM, you must perform a SAVE after the IP instruction. Otherwise, the old values will be restored once power is cycled.

Syntax Example

```
PRINT IOS 20                'Show defualt settings
IOS 20=0,1,1,1,0,0          'Change ios settings
PRINT IOS 20                'Show changes
IP                          'Clear all
PRINT IOS 20                'Show cleared to default
```

Related Commands

ALL, VARS, FLAGS, IOS

JOGS Setup Variable	Jog Speed Variable

Usage Example	Unit	Range	Default	Binary Mode Opcode Hex (Decimal)
JOGS=<speed>	User Units/sec	± .0000000000000001 to 9,999,999,999,999,999	256000.000	83h (131)

Notes

Speed at which the motor should move when a jog motion is performed.

The jog motion is performed in response to an input which is assigned the Jog Plus or Jog Minus type. When inputs have been designated with these types via IOS variables, the closure of the Jog Plus input causes the motor to move in the positive direction at the speed specified by JOGS. Similarly, the closure of the Jog Minus input causes the motor to move in the negative direction at the speed specified by JOGS.

Related Commands

MUNIT, IOS

JSC Setup Variable	Joystick Center Position Variable			
Usage Example	**Unit**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
JCS=<num>	AUNIT	0 - 4095	2048 (AUNIT=1)	84h (132)

Notes The JSC variable supports the Analog Input/Joystick Interface module and is updated automatically by means of the IJSC instruction, or can be updated manually as shown above.

Related Commands IJSC, JSDB, JSFS, JSE

JSDB Setup Variable	Joystick Deadband Variable			
Usage Example	**Unit**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
JSDB=<num>	AUNIT		10 (AUNIT=1)	85h (133)

Notes The JSDB variable supports the Analog Input/Joystick Interface module and is updated automatically by means of the IJSC instruction, or can be updated manually as shown above.

Related Commands IJSC, JSC, JSFS, JSE

JSE Setup Flag	Joystick Enable/Disable Flag		
Usage Example	**State**	**Default**	**Binary Mode Opcode Hex (Decimal)**
JSE = <flg>	<flg> = FALSE (0): Disabled. <flg> = TRUE (1): Enabled.	FALSE (0)	D0h (208)

Notes The JSE flag enables/disable joystick (velocity) mode for the MicroLYNX Analog Input/Joystick Module.

Related Commands IJSC, JSC, JSFS, JSDB

JSFS Setup Variable	Joystick Full Scale Variable			
Usage Example	**Unit**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
JSFS=<num>	AUNIT		2038	B5h (133)

Notes The JSFS variable supports the Analog Input/Joystick Interface module and is updated automatically by means of the IJSC instruction, or can be updated manually as shown above.

Related Commands IJSC, JSC, JSDB, JSE

LBL Program Mode Instruction	Label Program/Subroutine Instruction	
Usage Example	**Parameter**	**Binary Mode Opcode Hex (Decimal)**
LBL = <name>	<name> = 1 - 8 Alphanumeric characters and underscore (_).	42h (66)

Notes

This instruction will label the address of a program or subroutine within a program.

The name of the label can be 1 to 8 alphanumeric characters in length. You may use the underscore (_) character in the name as well. The first character of a label cannot be a numeral.

Subroutine calls, branches, program execution, events (trip) and break points can refer to the label instead of the address.

Syntax Example

```
PGM 100                 'Begin program at address line 100 of memory
LBL MY_PGM              'Name the program MY_PGM
PRINT "This is my program"
END                     'End the program
```

Related Commands

CALL, BR, EXEC, TI[1-4], TT[1-4],TP[1-4], TV BREAK

LDCLT Setup Variable	Limit Deceleration Type Variable		
Usage Example	**Parameters**	**Default**	**Binary Mode Opcode Hex (Decimal)**
LDCLT=<param>	<param>=0: User Defined <param>=1: Linear <param>=2: Triangle S-Curve <param>=3: Parabolic <param>=4: Sinusoidal S-Cuve	1 - Linear	86h (134)

Notes

The LDCLT Variable defines the type of curve that will be used to build deceleration when a limit has been hit. The deceleration profiles are defined as follows:

0 – User defined deceleration profile. This will follow the user defined points in the ACLTBL (acceleration table) for the acceleration profile.
1 – Constant (linear) deceleration.
2 – Triangle S-Curve profile.
3 – Parabolic profile.
4 – Sinusoidal S-Curve profile.

See DCLT in this section for an graphic example of deceleration types.

Comparison of Deceleration Types:

1 – Constant smooth (linear) deceleration from initial to max velocity.
2 – Triangle S-Curve profile.
3 – The Parabolic profile best utilizes the speed torque characteristics of a stepper motor since the highest acceleration takes place at low speed. It will, however, be the profile that results in the maximum jerk and is not recommended for applications requiring smooth starting and stopping. Such applications would include those that pull a material or move liquid.
4 – The Sinusoidal S-Curve profile is very similar to #3, the triangle S-Curve. The main difference is that it has less jerk when starting or stopping.

Related Commands

DECL, ACLTBL

LDECL Setup Variable	Limit Deceleration Variable			
Usage Example	**Unit**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
LDECL=<num>	User Units per second^2	±.0000000000000001 to ±9,999,999,999,999,999	1.000000.000	87h (135)

Notes

The LDECL Variable sets the peak deceleration that will be reached by the LYNX or MicroLYNX when a limit is reached in user units per second^2. If the user units have not been set then the value is in clock pulses per second^2.

The actual deceleration profile is maintained by the LDCLT variable. The value given by LDECL sets the maximum deceleration that the Control Module will reach.

Related Commands MUNIT, LDCLT

LIMSTP Setup Flag	Limit Stop Flag		
Usage Example	**Status**	**Default**	**Binary Mode Opcode Hex (Decimal)**
LIMSTP=<flg>	<flg> = FALSE (0): Do not stop program. <flg> = TRUE (1): Stop program.	FALSE (0)	D2h (210)

Notes

The Limit Stop Flag specifies whether (1) or not (0) the program should be stopped automatically when a limit is reached.

Regardless of the state of LIMSTP, an error is generated when a limit is reached. If LIMSTP is FALSE (0) when a limit is reached, the program will continue to run. In this case, the user should write code to take care of stopping the axis in the routine that is executed ONERror. This gives the user flexibility in dealing with how motion should be stopped when a limit is reached.

Related Commands ONER

LIST Immediate Mode Instruction	List Stored Program Space Instruction	
Usage Example	**Parameter**	**Binary Mode Opcode Hex (Decimal)**
LIST = <lbl/addr>, <flg>	<lbl/addr> = Starting label or address <flg> = 0: List through first END. <flg> = 1: List through end of program space.	43h (67)

Notes

If LIST is issued with no starting address specified, then the entire program space is reported to the host. If it is issued with a starting address and no stop flag or a stop flag of 0, then the program space is listed from the specified starting address to the first END that is encountered. Finally, if it is issued with a starting address and a stop flag of 1, then the program space is listed starting from the specified address and continuing until the end of the program space.

LOGO Setup Flag	Sign On Banner Enable/Disable Flag		
Usage Example	**Status**	**Default**	**Binary Mode Opcode Hex (Decimal)**
LOGO=<flg>	<flg> = FALSE (0): Disabled. <flg> = TRUE (1): Enabled.	TRUE (1)	D3h (211)

Notes This simply controls whether or not when the LYNX Product powers up a sign-on banner is echoed out the serial port. This banner consists of copyright and version information.

MAC Setup Variable	Motor Acceleration Current Setting Variable			
Usage Example	**Unit**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
MAC=<percent>	Percent	0 - 100	25	88h (136)

Notes This variable controls the percent of driver output current to be used when the axis is accelerating. See the section on current control in the part of this document pertaining to your product for more information. Figure 4:4 illustrates the relationship between the current control variables.

Related Commands MRC, MHC, PMHCC

MHC Setup Variable	Motor Holding Current Setting Variable			
Usage Example	**Unit**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
MHC=<percent>	Percent	0 - 100	5	8Ah (138)

Notes This variable controls the percent of driver output current to be used when the axis is between moves. See the section on current control in the part of this document pertaining to your product for more information.

Related Commands MRC, MAC, PMHCC

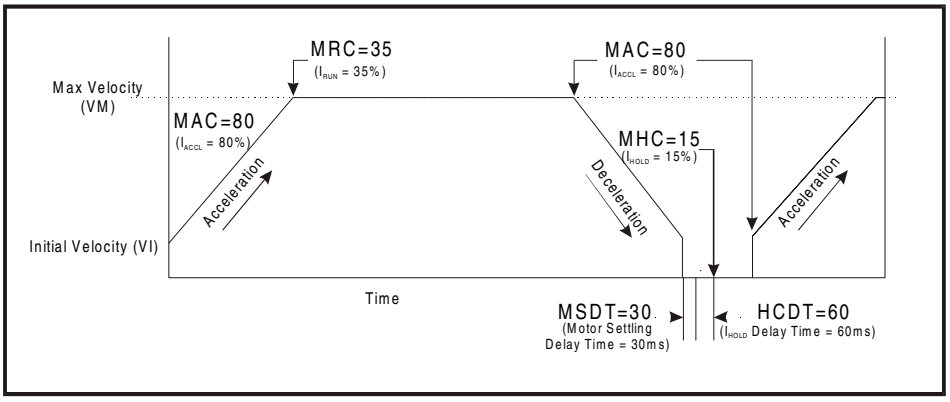


Figure 4.6: The Relationship Between the Current Control Variables

MOVA Immediate/Program Instruction	Move To Absolute Position Instruction		
Usage Example	**Parameters**	**Default**	**Binary Mode Opcode Hex (Decimal)**
MOVA =<position>, <mode>	<position> = ± Absolute position. <mode> = 0: Decelerate to position and stop. <mode> = 1: Do not decelerate, move part of profile.	Mode 0	44h (68)

Notes

There are two parameters to the MOVA instruction. The first specifies the absolute position to which the axis should move. The second specifies the mode of the movement. If mode = 0 then the axis should just stop when the specified position is reached. If mode = 1 then the motion is part of a profile and the motor should not decelerate to the specified position. In this case, it is expected that a new motion will take place immediately after the position is reached, so the motion continues at the final speed. Note that if mode is not specified, it is the same as having specified a mode of 0.

If MUNIT has been specified, then the position should be given in user units. Otherwise, the position should be specified in clock pulses.

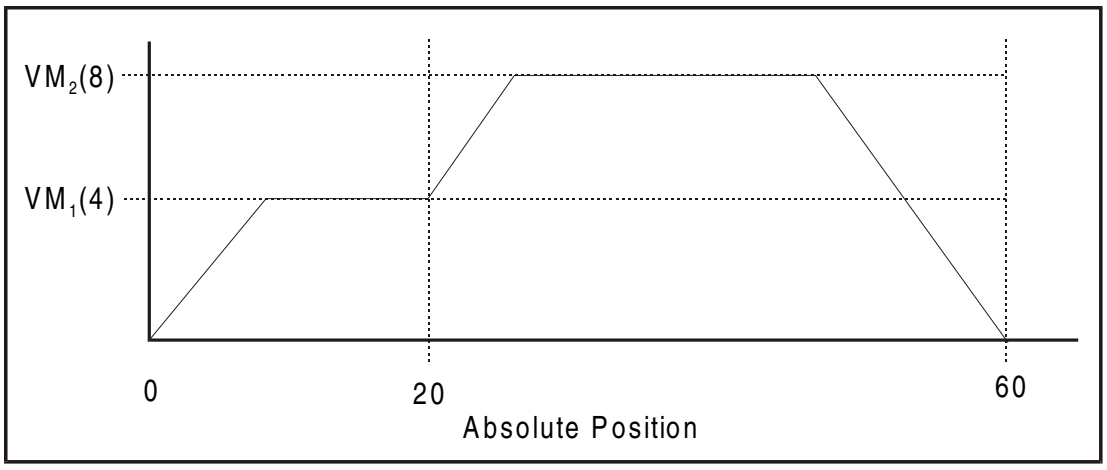


Figure 4.7: MOVA Instruction, Mode 1

Syntax Example

This example will use the MOVA instruction to create the profile shown below. Ensure that your start position is set to absolute 0.

```
POS = 0                 'Set Position to 0
PGM 100                 'Start program at address 100
LBL MOVADEMO            'MOVADEMO program
VM = 4                  'Maximum velocity set to 4 user units/sec for move 1
MOVA 20,1               'Index to absolute position 20, do not decelerate
HOLD 0                  'Suspend program execution until completion of position change
VM = 8                  'Maximum velocity set to 8 user units/sec for move 2
MOVA 60                 'Index to absolute position 60, decelerate and stop
END                     'End program
PGM                     'Return to immediate mode
```

Related Commands

VI, VM, ACL, ACLT, DCL, DCLT

MOVR Immediate/Program Instruction	Move To Relative Position Instruction		
Usage Example	**Parameters**	**Default**	**Binary Mode Opcode Hex (Decimal)**
MOVR =<position>, <mode>	<position> = ± Relative position. <mode> = 0: Decelerate to position and stop. <mode> = 1: Do not decelerate, move part of profile.	Mode 0	45h (69)

Notes

The primary difference between MOVA and MOVR is that where MOVA indexes to a position, MOVR will index a distance from the current position.

There are two parameters to the MOVR instruction. The first specifies the relative position to which the axis should move. The second specifies the mode of the movement. If mode = 0 then the axis should just stop when the specified position is reached. If mode = 1 then the motion is part of a profile and the motor should not decelerate to the specified position. In this case, it is expected that a new motion will take place immediately after the position is reached, so the motion continues at the final speed. Note that if mode is not specified, it is the same as having specified a mode of 0.

If MUNIT has been specified, then the position should be given in user units. Otherwise, the position should be specified in clock pulses.

Syntax Example

```
MOVR -10      'Specify a relative move of 10 user units in the -
direction
```

A profile within a program can be performed in the same fashion as the example given in the MOVA example. If MOVR is used, then the motion would start from the current location.

Related Commands VI, VM, ACL, ACLT, DCL, DCLT

MRC Setup Variable	Motor Run Current Setting Variable			
Usage Example	**Unit**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
MRC=<percent>	Percent	0 - 100	25	8Ch (140)

Notes

This variable controls the percent of driver output current to be used when the axis is at velocity. See the section on current control in the part of this document pertaining to your product for more information. Figure 4:4 illustrates the relationship between the current control variables.

Related Commands MAC, MHC, PMHCC

MSDT Setup Variable	Motor Settling Delay Time Variable			
Usage Example	**Unit**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
MSDT=<time>	Time in milliseconds.	0 - 65,535	0	8Eh (142)

Notes

Specifies the motor settling delay time. This is the time between moves if consecutive motions are executed. The PCHG and MVG flags are not cleared until the settling time has elapsed, so the settling time is included in the move time and will effect the HOLD command.

Related Commands PCHG, MVG, HOLD

MSEL Setup Variable	Motor Resolution Select Variable		
Usage Example	**Parameters**	**Default**	**Binary Mode Opcode Hex (Decimal)**
MSEL=<param>	See Table Below	256	91h (145)

Notes

The MSEL variable controls the microstep resolution of the MicroLYNX or driver module. There are 14 parameters that can be used with this variable, 8 binary and 6 decimal. The table below illustrates the parameter settings and their associated resolutions for a 1.8° stepper motor.

If using a motor with a step angle other than 1.8°, the microsteps/rev resolution will change with the step angle of the motor.

For example: a .45° step angle motor (800 full steps/rev) with MSEL variable set to MSEL=16, or 16 microsteps/step will have a resolution of 12,800 microsteps/rev.

The MSEL parameters given in the table below are the only valid parameters that will be accepted by the LYNX.

Microstep Resolution Settings	
MSEL Parameter (Microsteps/Step)	**Microsteps/Rev**
Binary Microstep Resolution Settings (1.8° Motor)	
2	400
4	800
8	1,600
16	3,200
32	6,400
64	12,800
128	25,600
256	51,200

Decimal Microstep Resolution Settings (1.8° Motor)	
5	1,000
10	2,000
25	5,000
50	10,000
125	25,000
250	50,000

MUNIT Setup Variable	Motor Units Variable			
Usage Example	**Unit**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
MUNIT=<num>	Clock Pulses per User Unit	±.0000000000000001 to ±9,999,999,999,999,999	1.000	91h (145)

Notes

Conversion factor for converting Clock pulses to user units. When the encoder is not enabled (EE = 0), POS will have the value of the scaled clock pulses. In other words, CTR1 / MUNIT will equal POS.

Syntax Example

If the encoder is enabled (EE = 1), then the value of EUNIT should be set to provide a conversion factor for encoder counts to user units.

Suppose the system will be moving a part in the linear axis in millimeters. Set the MUNIT to indicate that there are 800 motor steps/rev and 20 motor steps per millimeter. Therefore 1 revolution of the motor will equal 40 millimeters of linear motion with a full step drive. (Please note that actual step count per user unit will depend upon such factors as leadscrew pitch, gearing, etc. Any numbers given here are strictly arbitrary.)

```
MUNIT = 20     'Set MUNIT variable to 20, user unit will be millimeters
```

In another example we will use a micro-stepping drive set to a micro-step resolution of 256 micro-steps/step driving a .45° stepping motor. A .45° motor will have 800 steps per revolution (360 ÷ .45). 800 steps multiplied by the micro-step resolution of 256 will give us 204,800 micro-steps, or clock pulses, per revolution. For the purpose of example we will say that 123,456 micro-steps will equal 1 inch of linear movement. We would then divide the number of micro-steps per rev by the number of micro-steps per inch. 204,800 ÷ 123456 = 1.66 inches per revolution. Using the math functions built into the LYNX we can express the MUNIT value as follows:

```
MUNIT = 204800/1.66          'Set inches as user unit.
MOVR 5                       'Index 5" relative to current position
```

Related Commands

EUNIT, POS, EE

MVG Read Only Status Flag	Moving Flag		
Usage Example	**Status**	**Default**	**Binary Mode Opcode Hex (Decimal)**
BR <lbl/addr> MVG BR <lbl/addr> !MVG PRINT MVG	MVG = FALSE (0): Motor is stationary. MVG = TRUE (1): Motor is moving.	FALSE (0)	D5h (213)

Notes

Read only status flag which is TRUE (1) whenever the motor is moving.

This flag is TRUE (1) whenever the motor is moving regardless of the type of move, point-to-point, jog or slew. When a profiled move is taking place, this flag does not become FALSE (0) until the motion command with mode 0 has completed.

Related Commands

PCHG, VCHG

NOP Program Mode Instruction	No Operation Instruction		
Usage Example	**Parameters**	**Default**	**Binary Mode Opcode Hex (Decimal)**
NOP			46h (70)

Notes

This instruction is used to fill up one byte of program space. It can be used if, in editing a program, there is a change in the line boundary that causes a gap in the program. It can also be used to leave space for future instructions. It is recommended, however, that programs are written to a file using a text editor and downloaded to the LYNX Product during debug. This will save a great deal of retyping during debug of the program.

Syntax Example

```
POS=0                 'Set position to 0
PGM 100               'Start program at address 100
       LBL NOPDEMO    'Label program "NOPDEMO"
       VM 4           'Max velocity 4 user units/sec
       NOP            'No operation
       MOVA 20,1      'Move absolute 20 user units, do not decelerate
       HOLD 0         'Suspend prog. until position change completes
       NOP            'No operation
       VM 8           'Max velocity 8 user units/sec
       MOVA 60        'Move absolute 60 user units, decelerate and halt
       NOP            'No operation
END                   'End Program
PGM                   'Return to immediate mode
```

ONER Immediate/Program Instruction	On Error Instruction		
Usage Example	**Parameters**	**Default**	**Binary Mode Opcode Hex (Decimal)**
ONER <lbl/addr>	<lbl/addr> = Subroutine to be called on error.		47h (71)

Notes

When an error occurs in a program or due to an immediate command, the specified subroutine is called. If a program was running when the fault occurs, once the error routine completes, program execution continues with the instruction after the one that caused the error. A program need not be running for the subroutine specified by ONER to run.

The error function is disabled by setting the address parameter of a subsequent ONER command to 0 or resetting the LYNX Product.

Syntax Example

Executing the following program will cause the above routine to be called when an error occurs, reporting the error to the host.

```
PGM 100                    'Start program at address 100
 LBL ERR_HND               'Label program "ERR_HND"
 PRINT "Error Number  ",ERROR,
 RET                       'Return from subroutine
 ONER ERR_HND              'On error, goto ERR_HND
END                        'End program
PGM                        'Return to immediate mode
```

If the error report is no longer desired it can be turned off as follows:

```
ONER 0
```

Related Commands

ERR, ERROR,

PARTY Setup Flag	Party Mode Enable/Disable Flag		
Usage Example	**Status**	**Default**	**Binary Mode Opcode Hex (Decimal)**
PARTY=<flg>	<flg> = FALSE (0): Disabled. <flg> = TRUE (1): Enabled.	FALSE (0)	D7h (215)

Notes

This flag should be set to TRUE (1) for LYNX/MicroLYNX systems that are used in a multidrop system (multiple LYNX Products connected on a common RS-485 channel.) It should be left as FALSE (0), the factory default, if a single unit is used.

While in PARTY mode, a LYNX system node will respond to commands that are addressed to its name (given in DN). In addition, it will respond to global commands which are specified by the "*" character in the name field. Also, if its QUED flag is TRUE, the system node will respond to commands which are specified by the "^" character in the name field. Also the controller will respond to ESC and ^C.

There is a hardware switch to enable party mode as well, but the software setting will override it.

Related Commands

HOST, QUED

PAUS Immediate Mode Instruction	Pause Program Execution Instruction		
Usage Example	**Parameters**	**Default**	**Binary Mode Opcode Hex (Decimal)**
PAUS			48h (72)

Notes

Suspends the executing program as well as any motion that is executing. The way the motion is suspended and resumed is determined by the value of PAUSM.

Immediate commands are allowed while the control module is paused.

To continue the program, use the RES instruction. To abort the program, use the END instruction.

Related Commands

RES, END, PAUSD, PAUSM

PAUSD Read Only Status Flag	Paused Program Execution Flag		
Usage Example	**Status**	**Default**	**Binary Mode Opcode Hex (Decimal)**
BR <lbl/addr>, PAUSD BR <lbl/addr>, !PAUSD PRINT PAUSD	PAUSD = FALSE (0): Program not paused. PAUSD = TRUE (1): Program paused.	FALSE (0)	D8h (216)

Notes

This read only status flag will indicate whether or not a program has been paused.

Related Commands

PAUS

PAUSM Setup Variable	Pause Mode Variable		
Usage Example	**Parameters**	**Default**	**Binary Mode Opcode Hex (Decimal)**
PAUSM=<mode>	<mode>=0: Normal deceleration, resume with RES. <mode>=1: LDECL deceleration, resume with RES. <mode>=2: Complete motion, stop with normal deceleration. <mode>=3: Complete motion, stop with LDECL deceleration. <mode>=4: Normal deceleration, no resume with RES. <mode>=5: LDECL deceleration, no resume with RES.	Mode 0	92h (146)

Notes

Determines how motion is stopped in response to the PAUS instruction and whether or not it is restarted in response to the RES instruction.

The following describes how motion is stopped and resumed for each value of PAUSM:

0 Interrupt motion with normal deceleration (DECL) and resume motion in response to a RES instruction.
1 Interrupt motion with the LDECL deceleration and resume motion in response to a RES instruction.
2 Complete the current motion stopping with the normal deceleration (DECL).
3 Complete the current motion stopping with the LDECL deceleration.
4 Interrupt motion with normal deceleration (DECL) but don't resume motion in response to a RES instruction.
5 Interrupt motion with the LDECL deceleration but don't resume motion in response to a RES instruction.

Related Commands PAUS, PAUSD, DECL, LDECL, RES

PCHG Read Only Status Flag	Position Change Flag		
Usage Example	**Status**	**Default**	**Binary Mode Opcode Hex (Decimal)**
BR <lbl/addr>, PCHG BR <lbl/addr>, !PCHG PRINT PCHG	PCHG = FALSE (0): Axis stationary. PCHG = TRUE (1): Axis is changing position.	FALSE (0)	D9h (217)

Notes

This read only status flag indicates whether or not the axis is trying to obtain a specified position.

This flag becomes TRUE when the axis is moving in a profile motion. It is FALSE when the axis is moving in a jog or slew motion and becomes FALSE after the specified position has been exceeded in a MOVA or MOVR instruction with mode = 1. When the motor is moving in jog or slew motion or after the position has been reached during a MOVA or MOVR instruction with mode = 1, MVG is TRUE.

See the example for MOVA where HOLD is used to wait until PCHG becomes FALSE before starting the second move in the profile.

PFMT Setup Variable	Print Format Variable		
Usage Example	**Parameters**	**Default**	**Binary Mode Opcode Hex (Decimal)**
PFMT=<num1>, <num2>, <param>	<num1>: Number of digits before the decimal (0 - 16). <num2>: Number of digits after the decimal (0 - 16). <param>=0: Spaces as placeholders. <param>=1: Zeros as placeholders. <param>=2: No padding.	10, 3, 2	93h (147)

Notes

The PFMT variable specifies the print format for numeric values.

There are three parameters with PFMT. The first specifies how many significant digits there will be before the decimal. The second specifies how many significant digits there will be after the decimal. And the third specifies the type of padding. Blank or 0 specifies padding with spaces, 1 specifies padding with zeros, and 2 specifies no padding.

There will be a total of 16 digits displayed so, if there are 10 digits specified to the left of the decimal, there can be at most 6 specified to the right.

Related Commands

PRINT, PRINT1, PRINT2

PGM Immediate Mode Instruction	Enter/Exit Program Mode Instruction
Usage Example	**Binary Mode Opcode Hex (Decimal)**
PGM <addr> (Enter program mode) PGM (Exit Program mode)	49h (73)

Notes

When starting program mode, you must specify at what address to enter the program instructions in the program space. Simply type “PGM” again when you have finished entering your program commands to go back to immediate mode.

While in program mode, blank lines are accepted as are tab characters. This allows the user to format a text file with a user for readability, and then download the program to the LYNX by transferring the text file in a program such as HyperTerminal. The example given below could be stored in a text file and downloaded. The lines preceded by an apostrophe (‘) are comments and will be ignored by the LYNX Product. When the program is listed, the tabs and blank lines will not show, but they are accepted by the control module for input.

PGM Keyword	Retrieve Program Keyword
Usage Example	**Binary Mode Opcode Hex (Decimal)**
GET PGM	94h (148)

Notes

Used with GET to signify that all the program space should be retrieved from nonvolatile memory (NVM).

Related Commands

GET

PME Setup Flag	Position Maintenance Enable/Disable Flag		
Usage Example	**Status**	**Default**	**Binary Mode Opcode Hex (Decimal)**
PME=<flg>	<flg> = FALSE (0): Disabled. <flg> = TRUE (1): Enabled.	FALSE (0)	DAh (218)

Notes Specifies whether the position maintenance function, which maintains position within a specified deadband, is enabled (1) or disabled (0). The default setting is (0) disabled. In order for position maintenance to be performed, the Encoder enable flag (EE) must also be set to TRUE (1).

Related Commands EE, EDB

PMHCC Setup Variable	Position Maintenance Holding Current Change Variable			
Usage Example	**Unit**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
PMHCC=<percent>	Percent	0 to 100	0	95h (149)

Notes This variable specifies the amount of current required to maintain position when position maintenance is enabled.

The value for PMHCC is a percentage, its range being from 0% to 100% and is added to MHC until MRC is reached. Thus, if MHC is set to 15%, and MRC is set to 50% then the effective range for PMHCC will be 15 - 50%.

Related Commands EE, EDB, PME, PMV, MUNIT, EUNIT

PMV Setup Variable	Position Maintenance Velocity Variable			
Usage Example	**Unit**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
PMV=<speed>	User Units per second	±.0000000000000001 to ±9,999,999,999,999,999	10240.000	96h (150)

Notes Velocity to be used during position maintenance repositioning. If EUNIT has been set, then the value of PMV should be specified in user units. Otherwise, the value is simply specified in clock pulses per second.

Related Commands EE, EDB, PME, MUNIT, EUNIT

POS Variable	Axis Position Variable			
Usage Example	**Unit**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
POS=<±position> PRINT POS BR <lbl/addr>, POS=<±position>	User Units	± .0000000000000001 to ± 9,999,999,999,999,999	0.000	97h (151)

Notes

Contains the current position of the axis in user units. If the encoder is disabled, the POS register contains the scaled information that has been sent to the drive. In other words, POS = CTR1/MUNIT. In this case, if the user changes POS, CTR1 is also modified.
If the encoder is enabled, the POS register contains the scale information that has been seen at the encoder. In other words, POS = CTR2/EUNIT. In this case, if the user changes POS, CTR1 and CTR2 are both modified.

Modifying POS in essence changes the frame of reference for the axis. POS will probably be set once during system set up to reference or "home" the system.

Related Commands CTR1, CTR2, EE, MUNIT, EUNIT, POSCAP

POSCAP Read Only Variable	Axis Position At Time Of Trip Variable			
Usage Example	**Unit**	**Response**	**Default**	**Binary Mode Opcode Hex (Decimal)**
PRINT POSCAP	User Units	± Position	0.000	8Bh (139)

Notes

The POSCAP variable is a read only variable that captures the value of POS when a trip is encountered.

Related Commands POS, TIx, TIEx, TPx, TPEx, TTx, TTEx, TTRx, TVx, TVEx

PRINT Immediate/Program Instruction	Print Instruction
Usage Example	**Binary Mode Opcode Hex (Decimal)**
PRINT <"text"> PRINT <var/flg> PRINT <"text">,<var/flg>	4Ah (74)

Notes

This instruction is used to output text and parameter value(s) to the host PC. Text should be enclosed in quotation marks while parameters (variables and flags) should not. Text strings and parameters which are to be output by the same PRINT instruction should be separated by commas. The information being output is followed by a carriage return and line feed unless a semicolon (;) is included at the end of the PRINT instruction to indicate that the cursor should remain on the same line. This is useful when the PRINT instruction is being used to output instructions preceding an INPUT instruction.

The DISP instruction may effect how the data is printed. In addition, the PFMT variable will determine the representation of numerical data.

There are several control characters that can be embedded in the print text:

\b	Causes the cursor to backspace one character.
\c	Embeds a Ctrl-C into the text string.
\e	Embeds an ESC character into the text string to allow ANSI video escape sequences.
\g	Causes the terminal to sound the bell.
\n	Causes a line feed with no carriage return.
\r	Causes a carriage return with no line feed to allow overwriting of the same line.
\t	Embeds a Tab in the text string.

NOTE: These control characters MUST be lower case!

Syntax Example

This example will print the velocity and position information for the user's review.

```
PRINT "Velocity = ", VEL, "   Position = ", POS
```

The following example will request that the user input information into a variable. The cursor will remain on the same line for the user to input the data.

```
VAR TURNS                    'Declare user variable "TURNS"
PGM 100                      'Start program at address 100
       LBL SAMPLE            'Label program "SAMPLE"
       PRINT "Specify the number of turns: ";
       INPUT TURNS           'Request user input for TURNS
END                          'End program
PGM                          'Return to immediate mode
```

Related Commands DISP, INPUT, INPUT1, INPUT2, PFMT, PRINT1, PRINT2

PRINT1

Immediate/Program Instruction

Print to LYNX COMM1 Instruction

Usage Example	Binary Mode Opcode Hex (Decimal)
PRINT1 <"text"> PRINT1 <var/flg> PRINT1 <"text">,<var/flg>	59h (89)

Notes This is an enhancement of the PRINT instruction in that it will only output the print string to LYNX COMM 1, otherwise it operates the same as the PRINT instruction.

Related Commands DISP, INPUT, INPUT1, INPUT2, PFMT, PRINT, PRINT2

PRINT2

Immediate/Program Instruction

Print to LYNX COMM2 Instruction

Usage Example	Binary Mode Opcode Hex (Decimal)
PRINT2 <"text"> PRINT2 <var/flg> PRINT2 <"text">,<var/flg>	5Ah (90)

Notes This is an enhancement of the PRINT instruction in that it will only output the print string to LYNX COMM 2, otherwise it operates the same as the PRINT instruction.

Related Commands DISP, INPUT, INPUT1, INPUT2, PFMT, PRINT, PRINT1

PRMPT Setup Variable	Specify Prompt Character Variable			
Usage Example	**Unit**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
PRMPT=<char/ascii>	Character or ASCII decimal value	32 to 254	> (ASCII 62)	98h (152)

Notes Specifies the character that is used by the LYNX Product for a prompt. Valid characters are ASCII characters represented by decimal values 32 – 254. (See ASCII table in Appendix A)

QUED Setup Flag	Queue LYNX Controller Flag		
Usage Example	**Status**	**Default**	**Binary Mode Opcode Hex (Decimal)**
QUED=<flg>	<flg> = FALSE (0): Disabled. <flg> = TRUE (1): Enabled.	FALSE (0)	DBh (219)

Notes This flag, when TRUE (1), will enable LYNX nodes in a PARTY system to be able to receive broadcast commands. A queued node (one with QUED = 1) will respond to instructions addressed to "^". This in effect allows the host PC to broadcast instructions to multiple nodes in the system.

Related Commands PARTY

RATIO Setup Variable	Ratio Mode Variable		
Usage Example	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
RATIO=<num>	<-2 to <2	1.000	99h (153)

Notes The RATIO variable is used when one or more secondary drives is following the primary drive. This is done by setting the ratio option of IOS for one or more high speed output pairs to TRUE (1) and then setting RATIOE to TRUE (1). The clock driving the secondary drive(s) will be ratioed to the one driving the primary drive by the RATIO specified.

I/O lines 11 and 12 typically will be used for the primary. I/O lines 13 and 14 can be used to ratio other external drives as well. This would be done by setting the lines up as clock outputs with the ratio option of the IOS set to TRUE (1).

Syntax Example In the following example we will set the secondary axis (in this case CLK3) to follow the primary axis (CLK1) at a ratio of ½. **NOTE: A Differential Digital I/O module is required to perform this function. (Or a Combination I/O module using I/O line pairs 13 and 14 to control the secondary axis.)**

```
IOS 15 = 5,1,1,0,2,1        'Set Diff I/O channel 15 to ratio
IOS 16 = 6,1,1,0,2,1        'Set Diff I/O channel 16 to ratio
RATIO = .5                  'Set ratio to one half
RATIOE = 1                  'Set ratio mode enable flag to true
```

Related Commands IOS, RATIOE, RATIOW

RATIOE Setup Flag	Ratio Mode Enable Flag		
Usage Example	**Status**	**Default**	**Binary Mode Opcode Hex (Decimal)**
RATIO=<flg>	<flg> = FALSE (0): Disabled <flg> = TRUE (1): Enabled	FALSE (0)	DCh (220)

Notes This flag, when TRUE (1), will enable ratio mode operation. Although setting a parameter of the IOS variable specifies ratio mode, this flag acts as a master enable of the mode. This allows the user to enable and disable the function without changing the I/O setup. In addition, if multiple drives are being ratioed, this allows them to be started simultaneously.

Related Commands IOS, RATIO, RATIOW

RATIOW Setup Variable	Ratio Mode Pulse Width Variable			
Usage Example	**Parameters**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
RATIOW=<num>	<num> = 0: Square wave. <num> = 1 - 254: Pulses in increments of 50ns.	0 - 254	0	9Ah (154)

Notes Pulse width for the step clock of the secondary channel(s) being used to drive the motor(s) in ratio mode. It should be noted that if a square wave pulse is selected here, the ratio will be ½ that specified. For instance, if a ratio of 1 is specified and RATIOW is set to 0, the ratio will actually be ½. Thus, if a square wave pulse is desired, the true range of ratio is -1 <= RATIO < 1.

Related Commands RATIOE, RATIO

RES Immediate Mode Instruction	Resume Program Execution Instruction
Usage Example	**Binary Mode Opcode Hex (Decimal)**
RES	4Ch (76)

Notes Resume the program and, if necessary, motion that was suspended by a PAUS instruction. The program is always resumed, but the motion may or may not be resumed depending on the value of PAUSM at the time the PAUS instruction was issued.

Related Commands PAUS, PAUSM

RET Program Mode Instruction	Return From Subroutine Instruction	
Usage Example		**Binary Mode Opcode Hex (Decimal)**
RET		4Dh (77)

Notes A RET statement is required at the end of the subroutine executed by a CALL instruction.

Syntax Example

```
VAR VAL=0                    'Declare user variable "VAL", set to 0
PGM 100                      'Start Program at address 100
      LBL MAIN_PRG           'Label program "MAIN_PRG"
      MOVR 51200             'Move relative 51,200 user units
      HOLD 2                 'Suspend program until motion completes
      CALL SUB_ROUT,VAL=1    'Call subroutine "SUB_ROUT" when VAL=1
      BR MAIN_PRG            'Unconditional branch to MAIN_PRG
      LBL SUB_ROUT           'Declare subroutine SUB_ROUT
      MOVR 51200*5           'Move relative 51,200 X 5 user units
      HOLD 2                 'Suspend program until motion completes
      RET                    'Return from subroutine
END                          'End program
PGM                          'Return to immediate mode
```

Related Commands CALL

RUN Program Mode Instruction	Run Background Task Instruction	
Usage Example		**Binary Mode Opcode Hex (Decimal)**
RUN <lbl/addr>		4Eh (78)

Notes The RUN instruction starts a background task to be run at a specified address. When the background task is started, the foreground and background task both execute sharing the LYNX Product's processor. The background task runs until a RET or END instruction is reached or until the end of code space is reached. It is good practice to end the task using the RET or END instruction.

Note that only one background task may be executing at any one time. If you execute a second RUN instruction before the first one has completed, unexpected results will occur.

Syntax Example The following code sample will run a background task that will enable or disable an output based on the position of the motor while a foreground task is indexing the motor. In this example assume a half/full step driver set to full step driving a 1.8° stepping motor. When executed, the motor will move 1 revolution, set the output 31, move back to position 0, clear the output, then repeat.

The Foreground Program:

```
PGM 10                      'Enter program at line 10
LBL TST_PGM                 'Name the program TST_PGM
       MUNIT = 200          'Set MUNIT so that 200 units = 1 Revolution
       IOS 21 = 0,1         'Set I/O line 21 to a user defined output
       POS = 0              'Set the position to 0
       RUN BACK             'Run the background program labeled BACK
LBL LOOP                    'Define Sub Loop
       MOVA 200             'Index to Absolute Position 200
       HOLD 2               'Suspend Prog. execution until move completes
       DELAY 2000           'Delay 2 seconds
       MOVA 0               'Index to Absolute Position 0
       HOLD 2               'Suspend Prog. execution until move completes
       DELAY 2000           'Delay 2 seconds
       BR LOOP              'Unconditional Branch to Sub LOOP
END
PGM
```

The Background Program:

```
PGM 200
LBL BACK                    'Define background task BACK
       IO 21 = 0            'Set I/O 21 to 0
LBL FULL                    'Declare subroutine FULL
       BR FULL, POS = 200   'Loop to sub FULL until POS = 200
       IO 21 = 1            'Set I/O 21 to 1
       DELAY 4              'Delay Prog. execution 4 msec
LBL ZERO                    'Declare subroutine ZERO
       BR ZERO, POS = 0     'Loop to sub ZERO until POS = 0
       IO 21 = 0            'Clear I/O 21
       DELAY 2              'Delay Prog. execution 4 msec
       BR BACK              'Unconditional branch to BACK
END
PGM
```

Related Commands RET, END, BKGD, BKGDA

SAVE Immediate/Program Instruction	Save Instruction
Usage Example	**Binary Mode Opcode Hex (Decimal)**
SAVE	4Fh (79)

Notes

Saves all variables, flags and programs currently in working memory (RAM) to nonvolatile memory (NVM). The previous values in NVM are completely overwritten with the new values. If necessary, the user can get back to factory default values using the IP instruction.

When the user modifies variables, flags and program space, they are changed in working memory (RAM) only. If the SAVE instruction is not executed before power is removed from the control module, all modifications to variables, flags and programs since the last SAVE will be lost.

Related Commands IP, SET, PGM

SER Read Only Variable	Serial Number Variable	
Usage Example		**Binary Mode Opcode Hex (Decimal)**
PRINT SER		9Bh (155)

Notes This read only variable can be used to display the LYNX Product's serial number. The value set is at the factory

SET Immediate/Program Instruction	Set Variable Or Flag Instruction	
Usage Example		**Binary Mode Opcode Hex (Decimal)**
SET <var/flg> =<val>		50h (80)

Notes Sets a variable or flag to a specified value. SET is an optional command. It can be left off when assigning a value to a flag or variable. For instance, if the user wants to SET ACCL to 5, this can be done using the SET instruction (SET ACCL = 5) or the instruction can be implied (ACCL = 5).

Syntax Example In the below syntax example you will notice that we did not type the SET command in front of the variable name. In the LYNX software, the SET is assumed when a variable or flag value is defined. Whenever a program is uploaded from the LYNX to a text file or LISTed to the terminal screen, the SET instruction will appear in front of the variables and/or flags that have been defined within the program.

```
RATIOW = 200  'Set ratio pulse width to 10µs
```

SLEW Immediate/Program Instruction	Slew Motor At Constant Velocity Instruction		
Usage Example	**Units**	**Modes**	**Binary Mode Opcode Hex (Decimal)**
SLEW <±speed>, <mode>	<±speed> = User Units/sec	<mode> = 0: Use acceleration ramp. <mode> = 1: Do not use acceleration ramp.	51h (81)

Notes When using the SLEW instruction, the user must at least give a velocity (sign indicates direction) at which the motor should run. The slew velocity will be based upon the value of MUNIT. In addition, the user can specify whether or not the acceleration ramp should be used to get to speed. If the second parameter is not specified or is given as 0, the acceleration ramp should be used to get to speed. If it is specified as 1, the slew rate should be reached by a step function without acceleration.

Syntax Example
```
SLEW .5, 1    'Slew the motor .5 user units/sec w/no acceleration ramp
```

Related Commands ACCL

SSTP Immediate/Program Instruction	Soft Stop Instruction		
Usage Example	**Modes**	**Default**	**Binary Mode Opcode Hex (Decimal)**
SSTP <mode>	<mode> = 0: Stop motion only, program continues to execute. <mode> = 1: Stop both motion and program.	Mode 0	52h (82)

Notes

Stop the current motion using the specified deceleration profile and optionally stop the program that is currently running. If SSTP is issued with no parameter or 0, only the motion is terminated. If, however, SSTP is issued with a parameter of 1, the motion and program are both terminated.

Syntax Example

The examples below illustrate the SSTP instruction being used in both modes:

MODE 0

```
PGM 100              'Start program at address 100
       LBL TST       'Label the program "TST"
       SLEW 100000   'Slew th motor at 100000 user units/sec
       DELAY 3000    'Delay 3 seconds
       SSTP 0        'Soft stop motion, continue executing program
       DELAY 2000    'Delay 2 seconds
       BR TST        'Unconditional branch to beginning of program
END
PGM
```

MODE 1

```
PGM 100              'Start program at address 100
       LBL TST       'Label the program "TST"
       SLEW 100000   'Slew th motor at 100000 user units/sec
       DELAY 3000    'Delay 3 seconds
       SSTP 1        'Soft stop motion, stop program
       DELAY 2000    'Delay 2 seconds
       BR TST        'Unconditional branch to beginning of program
END
PGM
```

STALL Read Only Status Flag	Axis Stalled Indicator Flag		
Usage Example	**Status**	**Default**	**Binary Mode Opcode Hex (Decimal)**
BR <lbl/addr>, STALL BR <lbl/addr>, !STALL PRINT STALL	STALL = FALSE (0): Not stalled. STALL = TRUE (1): Axis stalled.	FALSE (0)	DEh (222)

Notes

Read only flag that indicates the motor has stalled. If the encoder is enabled (EE = 1) and the encoder "falls behind" the motor more than the specified factor, STLF, a STALL is indicated. If STLDE is also enabled (1), then the motor will be stopped when a STALL is detected.

Related Commands

EE, STLDE, STLF

STATS Keyword	Retrieve Status Flags Keyword	
Usage Example		**Binary Mode Opcode Hex (Decimal)**
PRINT STATS		9Ch (156)

Notes Used with the PRINT instruction to print values of the status flags only. The status flags are ACL, BKGD, BSY, DCL, ERR, HELD, MVG, PAUSD, PCHG, STALL, STK, VCHG.

Related Commands PRINT

STEPW Setup Variable	Step Pulse Width Variable			
Usage Example	**Parameters**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
STEPW=<num>	<num> = 0: Square wave. <num> = 1 - 254: Pulses in increments of 50ns.	0 - 254	0	9Dh (157)

Notes Step pulse width for the primary axis.

STK Read Only Status Flag	Subroutine Stack Fault Flag		
Usage Example	**Status**	**Default**	**Binary Mode Opcode Hex (Decimal)**
BR <lbl/addr>, STK BR <lbl/addr>, !STK PRINT STK	STK = FALSE (0): No fault. STK = TRUE (1): Stack overflow or underflow fault.	FALSE (0)	DFh (223)

Notes This is a read only flag that indicates a stack overflow or underflow.

STLDE Setup Flag	Stall Detect Enable/Disable Flag		
Usage Example	**Status**	**Default**	**Binary Mode Opcode Hex (Decimal)**
STLDE=<flg>	<flg> = FALSE (0): Disable. <flg> = TRUE (1): Enable.	FALSE (0)	E0h (224)

Notes If the encoder is enabled (EE = 1) and the encoder "falls behind" the motor more than the specified factor, STLF, a STALL is indicated. If STLDE is also enabled (1), then the motor will be stopped when a STALL is detected. EE is the master encoder enable - unless it is TRUE (1), nothing happens when STLDE becomes TRUE (1).

Related Commands EE, STALL, STLF, STLDM

STLDM Setup Variable	Stall Detection Mode Variable		
Usage Example	**Parameters**	**Default**	**Binary Mode Opcode Hex (Decimal)**
STLDM=<mode>	<mode> = 0: Stop motor when detecting a stall. <mode> = 1: Do not stop motor when detecting a stall.	Mode 0	89h (137)

Notes This variable sets the mode for stall detection.

Related Commands EE, STALL, STLF, STLDE

STLF Setup Variable	Stall Factor Variable			
Usage Example	**Unit**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
STLF=<num>	User Units	± .0000000000000001 to ± 9,999,999,999,999,999	10.000	9Eh (158)

Notes If the encoder is enabled (EE = 1) and the encoder "falls behind" the motor more than the specified factor, a STALL is indicated. If STLDE is also enabled (1), then the motor will be stopped when a STALL is detected.

Related Commands EE, STALL, STLDE

TI1, TI2, TI3, TI4 Setup Variables	Trip On Input Variables **FORMERLY IT<x>**		
Usage Example	**Parameters**	**Default**	**Binary Mode Opcodes Hex (Decimal)**
TI<x>=<input>, <lbl/addr>, <output>	<x> = 1 - 4 <input> = Input line used for trip. <lbl/addr> = Subroutine invoked on trip. <output> = Output set TRUE on trip.	0, 0, 0	TI1 = 7Fh (127) TI2 = 80h (128) TI3 = 81h (129) TI4 = 82h (130)

Notes Sets up an input event (trip) for the specified input. There are three parameters for the TI variables. The first specifies which input line should cause the event. The second specifies the address of the subroutine that should be executed when the input is seen. The third optional parameter specifies the output line to be set TRUE when the input trip is seen.

The input used should be a user input or one of the limit or home inputs. Note that the GO input automatically looks for a subroutine at address 1 and if there is valid code there it starts execution from address 1.

The TIE flag for the appropriate event number must be enabled for the event to be recognized.

Related Commands TIE1, TIE2, TIE3, TIE4, IOS

TIE1, TIE2, TIE3, TIE4 Setup Flags	Trip On Input Enable/Disable Flags **FORMERLY ITE<x>**		
Usage Example	**Status**	**Default**	**Binary Mode Opcode Hex (Decimal)**
TIE<x>=<flg>	<x> = 1 - 4 <flg> = FALSE (0): Disable. <flg> = TRUE (1): Enable.	FALSE (0)	TIE1 = CCh (204) TIE2 = CDh (205) TIE3 = CEh (206) TIE4 = CFh (207)

Notes Enables the corresponding event trip. Note the the input trips are disabled when the LYNX/ MicroLYNX encounters an END or RET statement.

Related Commands TI1, TI2, TI3, TI4

TP1, TP2, TP3, TP4 Setup Variables	Trip On Position Variables		
Usage Example	**Parameters**	**Default**	**Binary Mode Opcodes Hex (Decimal)**
TP<x>=<pos>, <lbl/addr>, <output>	<x> = 1 - 4 <pos> = ± Position in user units. <lbl/addr> = Subroutine invoked on trip. <output> = Output set TRUE on trip.	0.000, 0, 0	TP1 = A3h (163) TP2 = A4h (164) TP3 = A5h (165) TP4 = A6h (166)

Notes There are three parameters for the TPx variables. The first specifies the position at which the specified subroutine should be executed. The second specifies the address of the subroutine that should be executed when the position is reached. The third optional parameter specifies an output to be set TRUE when the trip is reached.

It should be noted that if EE is TRUE (1), in order to use TP3 and TP4 as encoder counts, the ENC input must be hard-wired to the EVENT input. The <pos> range is ±.0000000000000001 to ±9,999,999,999,999,999 user units based on the value of MUNIT.

Related Commands TPE1, TPE2, TPE3, TPE4, MUNIT

TPE1, TPE2, TPE3, TPE4 Setup Flags	Trip On Position Enable/Disable Flags		
Usage Example	**Status**	**Default**	**Binary Mode Opcode Hex (Decimal)**
TPE<x>=<flg>	<x> = 1 - 4 <flg> = FALSE (0): Disable. <flg> = TRUE (1): Enable.	FALSE (0)	TPE1 = E9h (233) TPE2 = EAh (234) TPE3 = EBh (235) TPE4 = ECh (236)

Notes These flags enable/disable the corresponding position event (trip).

Related Commands TP1, TP2, TP3, TP4

TT1, TT2, TT3, TT4 Setup Variables	Trip On Timer Variables FORMERLY TI<x>		
Usage Example	**Parameters**	**Default**	**Binary Mode Opcodes Hex (Decimal)**
TT<x>=<time>, <lbl/addr>, <output>	<x> = 1 - 4 <time> = Time in milliseconds (0 - 65,535). <lbl/addr> = Subroutine invoked on trip. <output> = Output set TRUE on trip.	0, 0, 0	TT1 = 9Fh (159) TT2 = A0h (160) TT3 = A1h (161) TT4 = A2h (162)

Notes There are three parameters for the TTx variables. The first specifies the period or time in milliseconds which should elapse before the event occurs. The second specifies the address of the subroutine that should be executed when the timer expires. The third optional parameter specifies an output to be set TRUE when the trip is reached.

TTRx specifies whether the associated event should be a one shot or repeated every time the specified period expires. TTEx must be enabled for the associated event to be recognized.

Related Commands TTE1, TTE2, TTE3, TTE4, TTR1, TTR2, TTR3, TTR4,

TTE1, TTE2, TTE3, TTE4 Setup Flags	Trip On Timer Enable/Disable Flags FORMERLY TIE<x>		
Usage Example	**Status**	**Default**	**Binary Mode Opcode Hex (Decimal)**
TTE<x>=<flg>	<x> = 1 - 4 <flg> = FALSE (0): Disable. <flg> = TRUE (1): Enable.	FALSE (0)	TTE1 = E1h (225) TTE2 = E2h (226) TTE3 = E3h (227) TTE4 = E4h (228)

Notes These flags enable the corresponding timer event (trip).

Related Commands TT1, TT2, TT3, TT4

TTR1, TTR2, TTR3, TTR4 Setup Flags	Trip On Timer Reload Flags FORMERLY TIR<x>		
Usage Example	**Status**	**Default**	**Binary Mode Opcode Hex (Decimal)**
TTR<x>=<flg>	<x> = 1 - 4 <flg> = FALSE (0): Do not repeat timer event. <flg> = TRUE (1): Repeat timer event.	FALSE (0)	TTR1 = E5h (229) TTR2 = E6h (230) TTR3 = E7h (231) TTR4 = E8h (232)

Notes TIRx specifies whether the associated event should be a one shot or repeated every time the specified period expires.

Related Commands TT1, TT2, TT3, TT4

TV Setup Variables	Trip On Velocity Variable FORMERLY VT		
Usage Example	**Parameters**	**Default**	**Binary Mode Opcodes Hex (Decimal)**
TV=<velocity>, <lbl/addr>, <output>	<velocity> = Velocity in user units/sec. <lbl/addr> = Subroutine invoked on trip. <output> = Output set TRUE on trip.	0.000, 0, 0	ACh (172)

Notes

There are three parameters for the VT variable. The first specifies the velocity at which the specified subroutine should be executed. The second specifies the address of the subroutine that should be executed when the velocity is reached. The optional third parameter specifies and output to be set TRUE when the trip is reached.

Once the trip has been set up, the specified subroutine is run when the velocity, VEL, passes through the velocity specified by vel. In other words, the subroutine will be called when the motor accelerates through the velocity and then again when it decelerates through it.

Note that the range of <velocity> is ±.0000000000000001 to ±9,999,999,999,999,999 user units based on the value of MUNIT.

Related Commands TVE, MUNIT

TVE Setup Flags	Trip On Velocity Enable/Disable Flag FORMERLY VTE		
Usage Example	**Status**	**Default**	**Binary Mode Opcode Hex (Decimal)**
TVE=<flg>	<flg> = FALSE (0): Disabled. <flg> = TRUE (1): Enabled.	FALSE (0)	EEh (238)

Notes This flags enables the corresponding velocity event (trip).

Related Commands TV

UFLGS Keyword	Report User Flags Keyword	
Usage Example		**Binary Mode Opcode Hex (Decimal)**
PRINT UFLGS		A7h (167)

Notes

This keyword is used with the PRINT instruction to report the state of all the user-defined flags which were created using the FLG instruction.
Returns:

G + Logic State = Global
L + Logic State = Local

Related Commands FLG

ULBLS Keyword	Report User Labels Keyword	
Usage Example		**Binary Mode Opcode Hex (Decimal)**
PRINT ULBLS		A8h (168)

Notes This keyword is is used with the PRINT instruction to report all the user-defined labels which were created using the LBL instruction.

Related Commands LBL

UVARS Keyword	Report User Variables Keyword	
Usage Example		**Binary Mode Opcode Hex (Decimal)**
PRINT UVARS		A9h (169)

Notes This keyword is used with the PRINT instruction to report the state of all the user defined variables which were created using the VAR instruction.
Returns:

G + Logic State = Global
L + Logic State = Local

Related Commands VAR

VAR Immediate/Program Instruction	Define User Variable Instruction	
Usage Example	**Parameters**	**Binary Mode Opcode Hex (Decimal)**
VAR <name> = <num>	<name> = 1 - 8 Alphanumeric characters and underscore (_). <num> = Some number.	54h (84)

Notes Defines a user variable that can contain numeric data. The name of the variable can be 1 to 8 alphanumeric characters in length. You may use the underscore (_) character in the name as well. The value of the variable can be initialized when it is defined. If it is not specifically initialized, it will have a value of 0 until it is set.

Variables can be “global” or “local”. A local variable is one that has been defined in a control module program while a global variable is defined in immediate mode. It should be noted that a local variable is not static, but is erased and declared again whenever the program is executed.

Syntax Example

```
PGM 100                 'Start program at address 100
        LBL TST         'Label program TST
        VAR MY_VAR      'Declare user variable MY_VAR, set to 1000 user units
        SLEW MY-VAR     'Slew the amount specified by MY-VAR
        BR TST          'Unconditional Branch to TST
END
PGM
```

Related Commands UVARS

VARS Keyword	Variables Keyword	
Usage Example		**Binary Mode Opcode Hex (Decimal)**
PRINT VARS GET VARS IP VARS		AAh (170)

Notes Used with the GET, IP and PRINT commands to specify that all variables should be retrieved from nonvolatile memory (NVM), set to their factory default values, or printed to the serial port, respectively. When used with the GET instruction, only system variable values are retrieved from NVM. When used with the IP instruction, only system variable values are set to the factory default parameters. In these cases, user defined variables are not affected. When used with the PRINT instruction, only variable values are echoed to the host computer.

Related Commands PRINT, IP, GET

VCHG Read Only Status Flag	Velocity Changing Flag		
Usage Example	**Status**	**Default**	**Binary Mode Opcode Hex (Decimal)**
BR <lbl/addr>, VCHG BR <lbl/addr>, !VCHG PRINT VCHG	VCHG = FALSE (0): Velocity constant. VCHG = TRUE (1): Velocity changing.	FALSE (0)	EDh (237)

Notes Read Only status flag indicates whether or not the axis is changing velocity. Will be TRUE (1) whenever the axis is accelerating or decelerating.

VEL Read Only Register Variable	Velocity Variable			
Usage Example	**Unit**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
PRINT VEL BR <lbl/addr>, VEL=<num> CALL <sub>, VEL =<num>	User Units/Sec.	± .0000000000000001 to ± 9,999,999,999,999,999	0.000	A8h (168)

Notes Register which contains the actual velocity of the axis in user units per second.

Related Commands EUNIT, MUNIT

VER Read Only Variable	Software Version Variable	
Usage Example	**Response**	**Binary Mode Opcode Hex (Decimal)**
PRINT VER	VER<chipsel#>=<board addr>, <version#>	A9h (169)

Notes This is a read only variable which will be changed only when the software is upgraded by using the upgrader program. It will print the software version of the LYNX Control Module, and the version of any add-on modules in the system. This list will not display when using the PRINT ALL, or PRINT VARS instruction, or if the communications mode selected is binary.

VI Setup Variable	Initial Velocity Variable			
Usage Example	**Unit**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
VI=<num>	User Units/Sec.	± .0000000000000001 to ± 9,999,999,999,999,999	102400.000	ADh (173)

Notes Initial velocity for the axis during a point-to-point motion. The factory default value is 102,400 clock pulses per second with a minimum value of 12,000 clock pulses per second when MUNIT = 1.

The initial velocity for a stepper should be set to avoid the low speed resonance frequency and must be set lower than the pull in torque of the motor.

Related Commands EUNIT, MUNIT

VM Setup Variable	Maximum Velocity Variable			
Usage Example	**Unit**	**Range**	**Default**	**Binary Mode Opcode Hex (Decimal)**
VM=<num>	User Units/Sec.	± .0000000000000001 to ± 9,999,999,999,999,999	768000.000	AEh (174)

Notes Maximum velocity for the axis during a point-to-point motion. The maximum velocity is the velocity that will be reached for any MOVA or MOVR, provided of course that the move is long enough for the axis to reach the velocity. When a motion occurs, the axis starts at velocity VI and accelerates using the specified acceleration profile until the velocity VM is reached.

Related Commands EUNIT, MUNIT

APPENDIX A

ASCII TABLE

Char	Decimal Value
Null	0
	1
	2
	3
	4
	5
	6
	7
	8
	9
	10
	11
	12
	13
	14
	15
	16
	17
	18
	19
¶	20
§	21
-	22
	23
	24
	25
	26
	27
	28
	29
	30
	31
	32
!	33
“	34
#	35
$	36
%	37
&	38
‘	39
(	40
)	41
*	42
+	43
,	44
-	45
.	46

Char	Decimal Value
/	47
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57
:	58
‘	59
<	60
=	61
>	62
?	63
@	64
A	65
B	66
C	67
D	68
E	69
F	70
G	71
H	72
I	73
J	74
K	75
L	76
M	77
N	78
O	79
P	80
Q	81
R	82
S	83
T	84
U	85
V	86
W	87
X	88
Y	89
Z	90
[	91
\	92
]	93

Char	Decimal Value
^	94
_	95
‘	96
a	97
b	98
c	99
d	100
e	101
f	102
g	103
h	104
i	105
j	106
k	107
l	108
m	109
n	110
o	111
p	112
q	113
r	114
s	115
t	116
u	117
v	118
w	119
x	120
y	121
z	122
{	123
|	124
}	125
~	126
	127
Ç	128
ü	129
é	130
â	131
ä	132
à	133
å	134
ç	135
ê	136
ë	137
è	138
ï	139
î	140

Char	Decimal Value
Ì	141
Ä	142
Å	143
É	144
æ	145
Æ	146
ô	147
ö	148
ò	149
û	150
ù	151
ÿ	152
Ö	153
Ü	154
¢	155
£	156
¥	157
	158
ƒ	159
á	160
í	161
ó	162
ú	163
ñ	164
Ñ	165
ª	166
º	167
¿	168
¬	169
¬	170
½	171
¼	172
¡	173
«	174
»	175
	176
	177
	178
	179
	180

Char	Decimal Value
	181
	182
	183
	184
	185
	186
	187
	188
	189
	190
	191
	192
	193
	194
	195
	196
	197
	198
	199
	200
	201
	202
	203
	204
	205
	206
	207
	208
	209
	210
	211
	212
	213
	214
	215
	216
	217
	218
	219
	220

Char	Decimal Value
	221
	222
	223
á	224
ß	225
Ã	226
ð	227
Ó	228
ó	229
µ	230
ô	231
Ö	232
È	233
Ù	234
ä	235
8	236
ö	237
å	238
	239
	240
±	241
	242
	243
	244
	245
÷	246
	247
°	248
·	249
·	250
	251
	252
²	253
	254
	255

APPENDIX B

Error Table

0 ..... no error

Hardware Errors

1018 IO MODULE NOT INSTALLED.
1019 LYNX CHECK SUM INCORRECT.
1100 FAULT/LIMIT DETECTED IN A CONNECTED DRIVE.
1101 FAULT IN DRIVE 1.
1102 FAULT IN DRIVE 2.
1103 FAULT IN DRIVE 3.
1105 DRIVE 1 FAULT AND TYPE CHANGED.
1106 DRIVE 2 FAULT AND TYPE CHANGED.
1107 DRIVE 3 FAULT AND TYPE CHANGED.
1109 DRIVE 1 MSEL COULD NOT BE SET.
1110 DRIVE 2 MSEL COULD NOT BE SET.
1111 DRIVE 3 MSEL COULD NOT BE SET.
1113 DRIVE 1 TYPE CHANGED, MSEL COULD NOT BE SET.
1114 DRIVE 2 TYPE CHANGED, MSEL COULD NOT BE SET.
1115 DRIVE 3 TYPE CHANGED, MSEL COULD NOT BE SET.
1117 DRIVE 1 FAULT, MSEL COULD NOT BE SET.
1118 DRIVE 2 FAULT, MSEL COULD NOT BE SET.
1119 DRIVE 3 FAULT, MSEL COULD NOT BE SET.
1121 DRIVE 1 FAULT, TYPE CHANGED, MSEL COULD NOT BE SET.
1122 DRIVE 2 FAULT, TYPE CHANGED, MSEL COULD NOT BE SET.
1123 DRIVE 3 FAULT, TYPE CHANGED, MSEL COULD NOT BE SET.
1125 HOLD IGNORED, MOTOR DISABLED.
1126 DRIVE 1 NOT AVAILABLE.
1127 DRIVE 2 NOT AVAILABLE.
1128 DRIVE 3 NOT AVAILABLE.
1130 DRIVE 1 TYPE CHANGED.
1131 DRIVE 2 TYPE CHANGED.
1132 DRIVE 3 TYPE CHANGED.
1134 ILLEGAL DRIVE NUMBER.
1201 SELECTED ANALOG BOARD NOT INSTALLED.
1202 ANALOG CHANNEL NUMBER NOT AVAILABLE.
1204 ANALOG OPTION NOT INSTALLED.
1205 ANALOG VALUE OUT OF RANGE, POSSIBLY DEFECTIVE BOARD.

I/O Errors

2001 FIOS FOUND NO (HOME) SWITCH.
2002 NOT IN FACTORY MODE.
2020 OUTPUT FAULT AT DIGITAL IO GROUP 20.
2021 OUTPUT FAULT AT DIGITAL IO LINE 21.
2022 OUTPUT FAULT AT DIGITAL IO LINE 22.
2023 OUTPUT FAULT AT DIGITAL IO LINE 23.
2024 OUTPUT FAULT AT DIGITAL IO LINE 24.
2025 OUTPUT FAULT AT DIGITAL IO LINE 25.
2026 OUTPUT FAULT AT DIGITAL IO LINE 26.
2030 OUTPUT FAULT AT DIGITAL IO GROUP 30.
2031 OUTPUT FAULT AT DIGITAL IO LINE 31.
2032 OUTPUT FAULT AT DIGITAL IO LINE 32.
2033 OUTPUT FAULT AT DIGITAL IO LINE 33.
2034 OUTPUT FAULT AT DIGITAL IO LINE 34.
2035 OUTPUT FAULT AT DIGITAL IO LINE 35.

2036 OUTPUT FAULT AT DIGITAL IO LINE 36.
2040 OUTPUT FAULT AT DIGITAL IO GROUP 40.
2041 OUTPUT FAULT AT DIGITAL IO LINE 41.
2042 OUTPUT FAULT AT DIGITAL IO LINE 42.
2043 OUTPUT FAULT AT DIGITAL IO LINE 43.
2044 OUTPUT FAULT AT DIGITAL IO LINE 44.
2045 OUTPUT FAULT AT DIGITAL IO LINE 45.
2046 OUTPUT FAULT AT DIGITAL IO LINE 46.
2050 OUTPUT FAULT AT DIGITAL IO GROUP 50.
2051 OUTPUT FAULT AT DIGITAL IO LINE 51.
2052 OUTPUT FAULT AT DIGITAL IO LINE 52.
2053 OUTPUT FAULT AT DIGITAL IO LINE 53.
2054 OUTPUT FAULT AT DIGITAL IO LINE 54.
2055 OUTPUT FAULT AT DIGITAL IO LINE 55.
2056 OUTPUT FAULT AT DIGITAL IO LINE 56.
2101 ANALOG RANGE NOT ALLOWED.
2102 ANALOG DESTINATION/SOURCE NOT ALLOWED.
2103 ANALOG DESTINATION/SOURCE ALREADY USED.
2104 INVALID ANALOG CHANNEL NUMBER.
2105 ANALOG LAW NOT ALLOWED.
2106 CAN'T ENABLE JOYSTICK WHILE IN MOTION OR CAN'T EXEC MOTION CMD WITH JOYSTICK ENABLED.
2200 CAN ERRORS: 1-6,11-16,21-26,31-36.

Clock Errors

3001 TRIED TO SET CLK TO NON CLOCK LINE OR WRONG LINE.
3002 CAN'T HAVE CLOCK TYPE APPLIED TO IT.
3003 CAN'T HAVE RATIO AND NO_CLK.
3004 CLK IO CAN'T BE SET FOR RATIO MODE.
3005 IN HALF-AXIS MODE.
3006 TRIED TO SET TO INPUT WHEN DRIVE CONNECTED.
3007 NO IO SET FOR INPUT + RATIO.

Syntax Errors

4001 INVALID IO NUMBER.
4002 TRIED TO WRITE GROUP TO NONUSER.
4003 TRIED TO WRITE TO A NON-USER LINE.
4004 TRIED TO WRITE TO AN INPUT.
4005 TRIED TO SET AN OUTPUT ONLY TO INPUT.
4006 TRIED TO SET AN INPUT ONLY TO OUTPUT.
4007 TRIED TO SET LINE TYPE TO LINE THAT CAN'T BE SET THAT WAY.
4008 NOT A VALID IO TYPE.
4009 IO TYPE SW. PREVIOUSLY DEFINED.
4010 FIND SW MUST BE SET AS INPUT.
4011 MORE THAN ONE IO SET FOR RATIO INPUT.
4012 ILLEGAL RUN/EXEC MODE.
4013 RECEIVED UNACCEPTIBLE COMMAND.
4014 ILLEGAL PAR IN "INPUT PAR" COMMAND.
4015 LABEL HAS TO BE TEXT.
4016 ILLEGAL DATA ENTERED IN PRINT FORMAT.
4017 NO DATA ENTERED, COMMAND IGNORED.
4018 ILLEGAL DRIVE NAME.
4019 ADDRESS DOESN'T POINT TO VALID INSTRUCTION.
4020 TRIED TO EXECUTE A BAD USER PROGRAM INSTRUCTION.
4021 ILLEGAL LINE NUMBER.
4022 MULTI LINE PRINTS NOT ALLOWED IN BINARY MODE.
4023 ILLEGAL HOLD TYPE.
4024 NOT ALLOWED IN IMMEDIATE MODE.

4025 AN INPUT IS ALREADY PENDING.
4026 SELECTED COMM, PORT2, CANNOT BE SEPERATELY SELECTED.
4027 LINE NUMBER NOT NEEDED.

Variable/Flag Errors

5001 ILLEGAL VARIABLE ENTERED.
5002 ILLEGAL FLAG ENTERED.
5003 ILLEGAL FLAG OR VARIABLE ENTERED.
5004 NO EQUAL IN: SET VARIABLE TO VALUE.
5005 ILLEGAL CHARACTER FOLLOWS DECLARATION OF VARIABLE OR FLAG.
5006 UNDEFINED USER VAR OR FLG.
5007 TRIED TO REDEFINE LBL/VAR/FLG.
5008 TRIED TO REDEFINE GBL/LCL LBL/VAR/FLG.
5009 INSTRUCTION/VARIABLE/FLAG NOT IMPLEMENTED IN THIS VERSION.
5010 VALUE OF LBL/VAR/FLG CHANGED - WARNING.
5011 FLAG IS READ ONLY.
5012 VARIABLE IS READ ONLY.
5013 CAN ONLY INIT ALL, VARS, FLAGS.
5016 CAN'T SET MULTI VARIABLES, READ ONLY.

Motion Errors

6001 REACHED PLUS LIMIT SW.
6002 REACHED MINUS LIMIT SW.
6003 TIME NEEDED TO MAKE MOVE LESS THAN 200USEC.
6004 NO DISTANCE FOR MOVE.

Encoder Errors

7001 STALL DETECTED.
7002 IMPROPER RATIO OF MUNIT TO EUNIT.
7003 MOVED OUT OF DEADBAND.

NVM Errors

8001 LABEL AREA FULL.
8002 SAVE FAILED.
8003 TRIED TO TAKE FROM EMPTY STACK.
8004 DATA NOT IN NVM.
8005 TRIED TO OVER FILL FOREGROUND STACK.
8006 TRIED TO SAVE WHILE MOTION IN PROGRESS.
8007 TRIED TO OVER FILL BACKGROUND STACK.
8008 BAD SECTOR IN PAGE 0 OF FLASH.
8009 BAD SECTOR IN PAGE 1 OF FLASH.
8010 BAD SECTOR IN PAGE 2 OF FLASH.
8011 BAD SECTOR IN PAGE 3 OF FLASH.

Out Of Range Errors

9001 IO FILTER OUT OF RANGE.
9002 IO GROUP OUT OF RANGE.
9003 PROGRAM ADDRESS OUT OF RANGE.
9004 RATIO OUT OF RANGE.
9005 DATA OUT OF RANGE FOR VARIABLE.
9006 PULSE WIDTH OUT OF RANGE.

9007 TOO MANY DIGITS SPECIFIED IN PRINT FORMAT.
9008 SUM OF ID AND FD EXCEEDS MAX NUMBER OF DIGITS.
9009 VALUE MUST BE POSITIVE ONLY.
9010 VM IS SET LESS THAN OR EQUAL TO VI.
9011 VI IS SET BELOW MIN_VELOCITY.
9012 MOVE DISTANCE TOO SHORT FOR PRESENT DECEL RATE.
9013 JOG SPEED LESS THAN MIN_VELOCITY.
9014 ANALOG INPUT NOT ALLOWED FOR DATA.
9015 COMM PORT OUT OF RANGE.

APPENDIX C

Factory Defaults

Variables

ACCL = 1000000.000
ACLT = 1
BAUD = 96
BKGDA = 0
BLM = 0
BLSH = 0.000
CTR1 = 0
CTR2 = 0
CTR3 = 0
DCLT = 1
DECL = 1000000.000
DISP = 0, 0, 0
DN = "!"
ECHO = 0
EDB = 2.000
ERROR = 0
EUNIT = 1.000
HAS = 1.000
IOF 10 = 0
IOF 20 = 7
IOF 30 = 7
IOF 40 = 7
IOF 50 = 7
IOS 11 = 1, 1, 1, 0, 2, 0
IOS 12 = 2, 1, 1, 0, 2, 0
IOS 13 = 3, 0, 1, 0, 1, 0
IOS 14 = 4, 0, 1, 0, 1, 0
IOS 15 = 5, 0, 1, 0, 1, 0
IOS 16 = 6, 0, 1, 0, 1, 0
IOS 17 = 0, 0, 1, 0, 0, 0
IOS 18 = 0, 0, 1, 0, 0, 0
IOS 21 = 0, 0, 1, 0, 0, 0
IOS 22 = 0, 0, 1, 0, 0, 0
IOS 23 = 0, 0, 1, 0, 0, 0
IOS 24 = 0, 0, 1, 0, 0, 0
IOS 25 = 0, 0, 1, 0, 0, 0
IOS 26 = 0, 0, 1, 0, 0, 0
IOS 31 = 0, 0, 1, 0, 0, 0
IOS 32 = 0, 0, 1, 0, 0, 0
IOS 33 = 0, 0, 1, 0, 0, 0
IOS 34 = 0, 0, 1, 0, 0, 0
IOS 35 = 0, 0, 1, 0, 0, 0
IOS 36 = 0, 0, 1, 0, 0, 0
IOS 41 = 0, 0, 1, 0, 0, 0
IOS 42 = 0, 0, 1, 0, 0, 0
IOS 43 = 0, 0, 1, 0, 0, 0
IOS 44 = 0, 0, 1, 0, 0, 0
IOS 45 = 0, 0, 1, 0, 0, 0
IOS 46 = 0, 0, 1, 0, 0, 0
IOS 51 = 0, 0, 1, 0, 0, 0
IOS 52 = 0, 0, 1, 0, 0, 0
IOS 53 = 0, 0, 1, 0, 0, 0
IOS 54 = 0, 0, 1, 0, 0, 0
IOS 55 = 0, 0, 1, 0, 0, 0
IOS 56 = 0, 0, 1, 0, 0, 0
IT1 = 0, 0
IT2 = 0, 0
IT3 = 0, 0
IT4 = 0, 0
JOGS = 256000.000
LDCLT = 1
LDECL = 1000000.000
MSDT = 0
MUNIT = 1.000
PAUSM = 0
PFMT = 10, 3, 2
PMV = 10240.000
POS = 0.000
PRMPT = ">"
RATIO = 1.000
RATIOW = 0
SER = ????????
STEPW = 0
STLF = 10.000
TI1 = 0, 0
TI2 = 0, 0
TI3 = 0, 0
TI4 = 0, 0
TP1 = 0.000, 0
TP2 = 0.000, 0
TP3 = 0.000, 0
TP4 = 0.000, 0
VEL = 0.000
VER = V.XXX
VI = 102400.000
VM = 768000.000
VT = 0.000, 0

Flags

ACL = FALSE
BIO = FALSE
BKGD = FALSE
BLE = FALSE
BSY = FALSE
CSE = FALSE
DCL = FALSE
EE = FALSE
ERR = FALSE
EVE = FALSE
GECHE = FALSE
HAE = FALSE
HELD = FALSE
HOST = TRUE
ITE1 = FALSE
ITE2 = FALSE
ITE3 = FALSE
ITE4 = FALSE
LIMSTP = FALSE
LOGO = TRUE
MVG = FALSE
PARTY = FALSE
PAUSD = FALSE
PCHG = FALSE
PME = FALSE
QUED = FALSE
RATIOE = FALSE
STALL = FALSE
STK = FALSE
STLDE = FALSE
TIE1 = FALSE
TIE2 = FALSE
TIE3 = FALSE
TIE4 = FALSE
TIR1 = FALSE
TIR2 = FALSE
TIR3 = FALSE
TIR4 = FALSE
TPE1 = FALSE
TPE2 = FALSE
TPE3 = FALSE
TPE4 = FALSE
VCHG = FALSE
VTE = FALSE

APPENDIX D

Establishing Communications using Windows95 Hyper Terminal

If your Host PC is equipped with the Windows9x or NT/2000 operating systems, you can create a new Hyper Terminal Setup by following these steps:

1. Select "Start"
2. Select "Programs"
3. Select "Accessories"
4. Select "Hyper Terminal"
5. Double Click on "Hypertrm.exe"
6. Then follow instructions:
   a) Enter name for this setup - "IMS Indexer" works well.
   And choose an Icon, then click "OK".
   b) Select Comm port to connect to. Should be a direct connection. Then click "OK".
   c) Now set the Comm Properties. (These are the factory default settings for the LYNX Product.
   These should be:

i)	Baud:	9600
ii)	Data Bits:	8
iii)	Parity:	None
iv)	Stop Bits:	1
v)	Flow Control:	None

Then click "OK".

You will now be in the Hyper Terminal you created. You will need to adjust its "Properties". Do this by clicking on the "Properties" Icon on the tool bar or by going to the "File" Menu item. Then select "Properties".

The "Properties" Setup window will show.

i) Select ANSI terminal.
ii) Select ASCII Setup.
   a) Set "Line delay" to 10 msec.
   b) Set "Character delay" to 0 msec.
   (if transfers hang up, increase 'Char. Delay.)
   c) Click "OK".

BE SURE TO SAVE YOUR SETUP.

Once you have established communication, the following will appear in you terminal window either on power-up or when the system has been reset using ^C (CTRL-C).

```
Program Copyright © 1996-1998 by:
Intelligent Motion Systems, Inc.
Marlborough, CT 06447
VER = 0.617  SER = A2698006
```

Page Intentionally Left Blank

TWENTY-FOUR MONTH LIMITED WARRANTY

Intelligent Motion Systems, Inc., warrants its products against defects in materials and workmanship for a period of 24 months from receipt by the end-user. During the warranty period, IMS will either, at its option, repair or replace Products which prove to be defective.

EXCLUSIONS

The above warranty shall not apply to defects resulting from: improper or inadequate handling by customer; improper or inadequate customer wiring; unauthorized modification or misuse; or operation outside of the electrical and/or environmental specifications for the Product.

OBTAINING WARRANTY SERVICE

To obtain warranty service, a returned material authorization number (RMA) must be obtained from customer service at (860) 295-6102 before returning product for service. Customer shall prepay shipping charges for Products returned to IMS for warranty service and IMS shall pay for return of Products to customer. However, customer shall pay all shipping charges, duties and taxes for Products returned to IMS from another country.

WARRANTY LIMITATIONS

IMS makes no other warranty, either expressed or implied, with respect to the Product. IMS specifically disclaims the implied warranties of merchantability and fitness for a particular purpose. Some jurisdictions do not allow limitations on how long an implied warranty lasts, so the above limitation or exclusion may not apply to you. However, any implied warranty of merchantability or fitness is limited to the 24-month duration of this written warranty.

EXCLUSIVE REMEDIES

If your Product should fail during the warranty period, call customer service at (860) 295-6102 to obtain a returned material authorization number (RMA) before returning product for service. Please include a written description of the problem along with contact name and address. Send failed product to: Intelligent Motion Systems, Inc., 370 N. Main St. Marlborough, Connecticut 06447. Also enclose information regarding the circumstances prior to Product failure.

LYNX Product Family Operating Instructions
LX-OM100-000
V03.10.2000